TUGAS AKHIR - KI141502

IMPLEMENTASI METODE *MULTI CRITERIA DECISION MAKING* MENGGUNAKAN *ANALYTIC HIERARCHY PROCESS* PADA *DOCKER CONTAINER* UNTUK OPTIMASI SUMBER DAYA *SERVER*

I DEWA PUTU ARDI NUSAWAN
NRP 5113100096

Dosen Pembimbing I
Royyana M. Ijtihadie, S.Kom., M.Kom., Ph.D

Dosen Pembimbing II
Bagus Jati Santoso, S.Kom., Ph.D

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya, 2017

(Halaman ini sengaja dikosongkan)

TUGAS AKHIR - KI141502

IMPLEMENTASI METODE *MULTI CRITERIA DECISION MAKING* MENGGUNAKAN *ANALYTIC HIERARCHY PROCESS* PADA *DOCKER CONTAINER* UNTUK OPTIMASI SUMBER DAYA *SERVER*

I DEWA PUTU ARDI NUSAWAN
NRP 5113100096

Dosen Pembimbing I
Royyana M. Ijtihadie, S.Kom., M.Kom., Ph.D

Dosen Pembimbing II
Bagus Jati Santoso, S.Kom., Ph.D

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya, 2017

(Halaman ini sengaja dikosongkan)

UNDERGRADUATE THESIS - KI141502

IMPLEMENTATION OF MULTI CRITERIA DECISION MAKING METHOD USING ANALYTIC HIERARCHY PROCESS ON DOCKER CONTAINER TO OPTIMIZE SERVER RESOURCES

I DEWA PUTU ARDI NUSAWAN
NRP 5113100096

Supervisor I
Royyana M. Ijtihadie, S.Kom., M.Kom., Ph.D

Supervisor II
Bagus Jati Santoso, S.Kom., Ph.D

Department of INFORMATICS
Faculty of Information Technology
Institut Teknologi Sepuluh Nopember
Surabaya, 2017

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

IMPLEMENTASI METODE *MULTI CRITERIA DECISION MAKING* MENGGUNAKAN *ANALYTIC HIERARCHY PROCESS* PADA *DOCKER CONTAINER* UNTUK OPTIMASI SUMBER DAYA *SERVER*

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Berbasis Jaringan
Program Studi S1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh :
I DEWA PUTU ARDI NUSAWAN
NRP: 5113100096

Disetujui oleh Dosen Pembimbing Tugas Akhir:

Royyana M. Ijtihadie, S.Kom, M.Kom, Ph.D. .....
NIP: 197708242006041001 (Pembimbing 1)

Bagus Jati Santoso, S.Kom., Ph.D .....
NIP: 051100116 (Pembimbing 2)

SURABAYA
Juli 2017

(Halaman ini sengaja dikosongkan)

IMPLEMENTASI METODE *MULTI CRITERIA DECISION MAKING* MENGGUNAKAN *ANALYTIC HIERARCHY PROCESS* PADA *DOCKER CONTAINER* UNTUK OPTIMASI SUMBER DAYA *SERVER*

Nama	:	**I DEWA PUTU ARDI NUSAWAN**
NRP	:	**5113100096**
Jurusan	:	**Teknik Informatika FTIf**
Pembimbing I	:	**Royyana M. Ijtihadie, S.Kom., M.Kom., Ph.D**
Pembimbing II	:	**Bagus Jati Santoso, S.Kom., Ph.D**

Abstrak

Saat ini di era *cloud computing*, membuat kebutuhan akses data ke *server* semakin meningkat. Dibutuhkan teknologi untuk mengoptimalkan *resource* yang tepat pada *server* agar dapat bekerja optimal. Penggunaan *docker container* semakin meningkat seiring dengan kebutuhan akan optimasi tersebut. *docker container* merupakan pilihan yang tepat sebagai pengganti *Virtual Machine*. Pada *container*, programmer dapat fokus pada pengengembangan aplikasi, serta dapat dipastikan bahwa aplikasi akan dapat berjalan pada *host* apapun. *container* menggunakan sumber daya *server*, dan setiap docker yang dijalankan pada *server* memakai sumber daya yang berbeda – beda tergantung dari berbagai faktor. Namun, *container* akan tetap berjalan walaupun tidak sedang dipakai. Untuk memaksimalkan sumber daya yang ada, diperlukan analisisa yang tepat untuk mematikan *container* yang sedang tidak digunakan. Berbagai parameter dapat dipakai sebagai acuan, seperti CPU yang digunakan, memori, dan waktu. Metode *Multi Criteria Decision Making (MCDM)* dapat membantu menentukan pilihan terbaik yang diperlukan. Dalam tugas akhir

ini, akan menggunakan salah satu metode MCDM yang sering dipakai, yaitu Analytic Hierarchy Process (AHP). Prosedur AHP telah diimplementasikan pada *Decision Support System (DSS)*, termasuk *data mining* dan *machine learning*, serta banyak aplikasi lainnya. Dengan menggunakan metode perhitungan AHP, dapat ditentukan *container* yang memiliki prioritas rendah hingga tinggi. Setelah ditentukan, maka *container* dengan prioritas rendah dapat dimatikan, sehingga sumber daya *server* dapat digunakan oleh *container* lain yang membutuhkan sumber daya lebih.

Kata-Kunci: *Docker, Container, MCDM, AHP*

IMPLEMENTATION OF MULTI CRITERIA DECISION MAKING METHOD USING ANALYTIC HIERARCHY PROCESS ON DOCKER CONTAINER TO OPTIMIZE SERVER RESOURCES

Name	:	**I DEWA PUTU ARDI NUSAWAN**
NRP	:	**5113100096**
Major	:	**Informatics FTIf**
Supervisor I	:	**Royyana M. Ijtihadie, S.Kom., M.Kom., Ph.D**
Supervisor II	:	**Bagus Jati Santoso, S.Kom., Ph.D**

Abstract

The era of cloud computing increases the need for data access to the server. It takes technology to optimize the right resources on the server in order to work optimally. The use of docker container increases along with the need for such optimization. Docker Container is the right choice as a substitute for Virtual Machine. In using Docker Container, programmers can focus on the development of the application, and it is certain that the application will be able to run on any host. The docker container uses resources from the server, and each docker that runs on the server uses different resources depending on various factors. However, the container will still run even when not in use. To maximize the available resources, proper analysis is needed to turn off the unused containers. Various parameters can be used as a reference, such as CPU use, memory, and the last accessed time. The Multi Criteria Decision Making (MCDM) method can help determine the best options needed. The algorithm used in this final project is the Analytic Hierarchy Process (AHP), as one of the commonly used algorithm in MCDM method. The AHP procedure has been implemented on

the Decision Support System (DSS), including data mining and machine learning, as well as in many other applications [1]. The use of AHP calculation method can determine the low or high priority of the container. Once specified, the low priority container can be turned off, so the server resources can be used by other containers that require more resources.

Keywords: *Docker, Container, MCDM, AHP*

KATA PENGANTAR

Segala puji bagi Ida Sang Hyang Widhi Wasa, yang telah melimpahkan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul **Implementasi Metode *Multi Criteria Decision Making* Menggunakan *Analytic Hierarchy Process* pada *Docker Container* untuk Optimasi Sumber Daya *Server***. Pengerjaan Tugas Akhir ini merupakan suatu kesempatan yang sangat baik bagi penulis. Dengan pengerjaan Tugas Akhir ini, penulis bisa belajar lebih banyak untuk memperdalam dan meningkatkan apa yang telah didapatkan penulis selama menempuh perkuliahan di Teknik Informatika ITS. Dengan Tugas Akhir ini penulis juga dapat menghasilkan suatu implementasi dari apa yang telah penulis pelajari. Selesainya Tugas Akhir ini tidak lepas dari bantuan dan dukungan beberapa pihak. Sehingga pada kesempatan ini penulis mengucapkan syukur dan terima kasih kepada:

1. Ida Sang Hyang Widhi Wasa atas anugerahnya yang tidak terkira kepada penulis.
2. Bapak Royyana Muslim Ijtihadie, S.Kom, M.Kom, PhD selaku pembimbing I yang telah membantu, membimbing, dan memotivasi penulis mulai dari pengerjaan proposal hingga terselesaikannya Tugas Akhir ini.
3. Bapak Bagus Jati Santoso, S.Kom., Ph.D selaku pembimbing II yang juga telah membantu, membimbing, dan memotivasi penulis mulai dari pengerjaan proposal hingga terselesaikannya Tugas Akhir ini.
4. Darlis Herumurti, S.Kom., M.Kom., selaku Kepala Jurusan Teknik Informatika ITS pada masa pengerjaan Tugas Akhir, Bapak Radityo Anggoro, S.Kom., M.Sc., selaku koordinator TA, dan segenap dosen Teknik Informatika yang telah memberikan ilmu dan pengalamannya.
5. Teman-teman Administrator LP, yakni Razi, Dewangga, Agha, Rona, Sani, Ine, Gideon, Udin, dan yang bersedia direpotkan, merepotkan dan menemani penulis dalam masa pengerjaan tugas akhir ini.

6. Teman - teman PPDB Surabaya 2016 serta PPDB Jawa Timur 2017 yang senantiasaa begadang bareng di lab badak dan lt4.
7. Widya Pramesti yang telah bersedia membantu menerjemahkan buku ini :)
8. Serta semua pihak yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan. Sehingga dengan kerendahan hati, penulis mengharapkan kritik dan saran dari pembaca untuk perbaikan ke depannya.

Surabaya, Juli 2017

I Dewa Putu Ardi Nusawan

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I

PENDAHULUAN

Pada bab ini akan dipaparkan mengenai garis besar Tugas Akhir yang meliputi latar belakang, tujuan, rumusan dan batasan permasalahan, metodologi pembuatan Tugas Akhir, dan sistematika penulisan.

1.1 Latar Belakang

Saat ini di era cloud computing, membuat kebutuhan akses data ke server semakin meningkat. Dibutuhkan teknologi untuk mengoptimalkan resource yang tepat pada server agar dapat bekerja optimal. Penggunaan *container* semakin meningkat seiring dengan kebutuhan akan optimasi tersebut. *Docker Container* merupakan pilihan yang tepat sebagai pengganti virtual machine. *Container* adalah sebuah *image* yang ringan, bisa berdiri sendiri, paket *executable* dari perangkat lunak yang mencakup semua yang diperlukan untuk menjalankan program didalamnya, yakni: kode, *runtime*, berbagai *tools* yang diperlukan oleh sistem, *library* sistem, dan berbagai pengaturan lainnya. Tersedia pada masing - masing aplikasi yang berbasiskan Linux dan Windows, software yang ter-*container* akan dapat berjalan dengan environment yang sama. *Container* mengisolasi program dari keadaannya diluar *container*, sebagai contohnya adalah perbedaan *environment* antara *development* dan *staging* dan membantu mengurangi konflik diantara team yang menjalankan software pada infrastruktur yang sama[3].

Pada *container*, programmer dapat fokus pada pengengembangan aplikasi, serta dapat dipastikan bahwa aplikasi akan dapat berjalan pada host apapun. *Container* menggunakan sumber daya server, dan setiap Docker yang dijalankan pada server memakai sumber daya yang berbeda – beda tergantung dari berbagai faktor.

Namun, *container* akan tetap berjalan walaupun tidak sedang

dipakai. Untuk memaksimalkan *resource* yang ada, diperlukan analisisa yang tepat untuk mematikan *container* yang sedang tidak digunakan. Berbagai parameter dapat dipakai sebagai acuan, seperti *CPU* yang digunakan, memori, dan waktu. Metode *Multi Criteria Decision Making (MCDM)* dapat membantu menentukan pilihan terbaik yang diperlukan.

Dalam tugas akhir ini, akan menggunakan salah satu metode MCDM yang sering dipakai, yaitu *Analytic Hierarchy Process (AHP)*. Prosedur AHP telah diimplementasikan pada *Decision Support System (DSS)*, termasuk *data mining* dan *machine learning*, serta banyak aplikasi lainnya. Dengan menggunakan metode perhitungan AHP, dapat ditentukan *container* yang memiliki prioritas rendah hingga tinggi. Setelah ditentukan, maka *container* dengan prioritas rendah dapat dimatikan, sehingga sumber daya server dapat digunakan oleh *container* lain yang membutuhkan sumber daya lebih.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam tugas akhir ini adalah sebagai berikut :

1. Bagaimana menerapkan skema MCDM dengan AHP untuk mengoptimalkan penggunaan sumber daya *server*?
2. Bagaimana penggunaan sumber daya *server* apakah terjadi pengoptimalan jika menggunakan metode pengambilan keputusan AHP dibandingkan dengan sistem yang berjalan tanpa pengoptimalan?

1.3 Batasan Masalah

Dari permasalahan yang telah diuraikan di atas, terdapat beberapa batasan masalah pada tugas akhir ini, yaitu:

1. Menggunakan *Docker Container*.

2. Menggunakan bahasa pemrograman Python sebagai *Middleware*.
3. Pengujian dilakukan menggunakan jaringan lokal Teknik Informatika FTIf ITS.
4. Menggunakan metode *AHP* untuk menentukan *container* yang harus dihentikan.
5. Hanya melakukan optimasi pada satu server fisik.

1.4 Tujuan

Tujuan dari pengerjaan Tugas Akhir ini adalah dapat menerapkan skema MCDM dengan metode AHP untuk mengoptimalkan sumber daya *server*.

1.5 Manfaat

Manfaat dari hasil pembuatan tugas akhir ini adalah dapat mengoptimalkan sumber daya *server* (RAM, *CPU*) agar lebih optimal dan efisien dalam menjalankan banyak *docker container*.

(Halaman ini sengaja dikosongkan)

BAB II

LANDASAN TEORI

2.1 *Analytical Hierarchy Proses* (AHP)

Analytic Hierarchy Process (AHP) diciptakan oleh Saaty (1980) dan sering disebut, secara eponim sebagai metode Saaty. Metode ini populer dan banyak digunakan, terutama dalam analisis militer, meskipada saat itu terbatas pada masalah militer. Sebenarnya, dalam bukunya Saaty menggambarkan berbagai aplikasi penggunaan metode AHP mulai dari pilihan sekolah untuk anaknya, hingga perencanaan sistem transportasi untuk Negara Sudan[1].

AHP menangani masalah seperti berikut: Terdapat berbagai pilihan pembeliannya: biaya*(expense)*, E; Operabilitas, O; Kehandalan*(reliability)*, R; Kemampuan beradaptasi untuk kegunaan lain, dan F; fleksibilitas. Produsen yang bersaing dengan peralatan tersebut telah menawarkan tiga pilihan, X, Y dan Z. Insinyur perusahaan telah melihat opsi ini dan memutuskan bahwa X murah dan mudah dioperasikan namun tidak terlalu dapat diandalkan. Dan tidak bisa dengan mudah disesuaikan dengan kegunaan lain. Y agak lebih mahal, cukup mudah dioperasikan, sangat bisa diandalkan tapi tidak terlalu mudah beradaptasi. Akhirnya, Z sangat mahal, tidak mudah dioperasikan, sedikit kurang dapat diandalkan dibanding Y namun diklaim oleh pabrikan untuk memiliki berbagai macam kegunaan alternatif[1].

Masing-masing X, Y dan Z akan memenuhi persyaratan perusahaan untuk memperluas jangkauan sehingga yang, secara keseluruhan, paling sesuai dengan kebutuhan perusahaan ini?[1]

2.1.1 Prinsip dasar AHP

AHP dan teknik perhitungannya secara singkat adalah membuat matriks yang mengekspresikan nilai relatif dari sekumpulan atribut. Misalnya, apa kepentingan relatif

pengelolaan perusahaan biaya peralatan ini dibandingkan dengan kemudahan pengoperasiannya? Mereka diminta untuk memilih apakah biaya jauh lebih penting, lebih penting, sama pentingnya, dan seterusnya menjadi sangat tidak penting, daripada pengoperasian. Masing-masing penilaian ini diberi nomor dalam skala. Skala yang umum (diadaptasi dari Saaty) digunakan dapat dilihat pada tabel 2.1:

Tabel 2.1: Scala Rating Saaty[1]

Tingkat kepentingan	**Definisi**	**Penjelasan**
1	Sama pentingnya	Dua faktor berkontribusi sama terhadap tujuan
3	Agak lebih penting	Pengalaman dan penilaian sedikit lebih menguntungkan dibanding yang lain
5	Jauh lebih penting	Pengalaman dan penilaian sangat mendukung satu dari lainnya
7	Sangat jauh lebih penting	Pengalaman dan penilaian sangat mendukung satu dari lainnya. Pentingnya ditunjukkan dalam praktek
9	Benar-benar lebih penting	Bukti yang mendukung satu dari yang lain adalah kemungkinan tertinggi
2, 4, 6, 8	Nilai intermediate	Ketika kompromi diperlukan

Asumsi dasar, tapi sangat masuk akal adalah bahwa jika atribut A benar-benar lebih penting daripada atribut B dan diberi nilai 9, maka B harus benar-benar kurang penting daripada A dan bernilai 1/9.

Perbandingan berpasangan ini dilakukan untuk semua faktor yang harus dipertimbangkan, biasanya tidak lebih dari 7, dan matriksnya selesai. Matriks adalah bentuk yang sangat khusus yang mendukung perhitungan yang kemudian terjadi (Saaty adalah seorang matematikawan yang sangat terhormat).

Langkah selanjutnya adalah perhitungan daftar bobot relatif, kepentingan, atau nilai, faktor-faktor, seperti biaya dan pengoperasian yang relevan dengan masalah yang dimaksud (secara teknis daftar ini disebut vektor eigen). Jika, mungkin, biaya sangat jauh lebih penting daripada operabilitas, maka pada interpretasi sederhana, peralatan yang lebih murah yang lebih digunakan.

Vektor eigen pertama memberi kepentingan relatif yang melekat pada persyaratan, seperti biaya dan keandalan namun mesin yang berbeda mempunyai biaya dan keandalan yang berbeda pula. Dengan demikian, matriks selanjutnya dapat dikembangkan untuk menunjukkan bagaimana X, Y dan Z masing-masing memenuhi kebutuhan perusahaan. Langkah terakhir adalah menggunakan perhitungan matriks standar untuk menghasilkan keseluruhan vektor dengan memberikan jawaban yang kita cari, yaitu manfaat relatif X, Y dan Z dibandingkan secara *vis-à-vis(France, face to face)* untuk menemukan pilihan terbaik yang sesuai dengan persyaratan perusahaan[1].

2.1.2 Kasus Penggunaan

2.1.2.1 Menyusun Matriks Perbandingan

Pertama, diberikan matriks awal untuk perbandingan berpasangan perusahaan di mana diagonal utama berisi entri 1,

karena setiap faktor sama pentingnya dengan dirinya sendiri. 2.2:

	E	O	R	F
E	1			
O		1		
R			1	
F				1

Tabel 2.2: Menyusun Matrik Diagonal Bernilai 1[1]

Perusahaan memutuskan bahwa O, operabilitas, sedikit lebih penting daripada biaya *(expense).* Diberi nilai 3 pada kolom O, E dan 1/3 di E, O. Mereka juga memutuskan bahwa biaya jauh lebih penting daripada keandalan *(reliability),* memberikan 5 di E, R dan 1/5 di R, E, seperti yang ditunjukkan pada tabel 2.3.

	E	O	R	F
E	1	1/3	5	
O	3	1		
R	1/5		1	
F				1

Tabel 2.3: Memasukkan Bobot Tahap 1[1]

Perusahaan juga menilai bahwa operabilitas, O, jauh lebih penting daripada keandalan (R) maka diberi nilai 5 pada kolom O, R dan penilaian yang sama dibuat mengenai kepentingan relatif F dengan R. Ini membentuk matriks yang disebut dengan Preference Matriks (OPM), seperti pada tabel 2.4:

	E	O	R	F
E	1	1/3	5	1
O	3	1	5	1
R	1/5	1/5	1	1/5
F	1	1	5	1

Tabel 2.4: Memasukkan Bobot Tahap 2[1]

Maka setelah dihitung nilai akar ke-n^{th} dan eigen vectornya, didapatkanlah seperti tabel 2.5:

	E	O	R	F	n^{th} root of product of values	Eigenvector
E	1	1/3	5	1	1.133368103	0.23149916
O	3	1	5	1	1.967989671	0.40197704
R	1/5	1/5	1	1/5	0.299069756	0.0610873
F	1	1	5	1	1.495348781	0.3054365
Totals					4.895776312	1.000

Tabel 2.5: Mengembangkan Bobot untuk Kriteria[1]

Eigenvector yang akan di sebut *Relative Value Vector (RVV)*, dihitung dengan metode standar didapatkan (0.232, 0.402, 0.061, 0.305). Keempat bilangan ini merupakan nilai relatif E, O, R dan F, dimana yang paling besar 0.402, sedangkan 0,305 menunjukkan bahwa mereka menyukai gagasan fleksibilitas; Dua angka yang tersisa menunjukkan bahwa mereka tidak terlalu khawatir dengan biaya dan tidak tertarik pada keandalan. Kelihatannya aneh untuk tidak tertarik pada keandalan tapi RVV menangkap semua faktor implisit dalam konteks keputusan. Mungkin, dalam kasus ini, mesin hanya akan digunakan sesekali sehingga akan ada banyak waktu untuk perbaikan jika dibutuhkan.

Sekarang beralih ke tiga mesin potensial, X, Y dan Z. Kita

sekarang memerlukan empat set perbandingan berpasangan tapi kali ini dalam hal seberapa baik X, Y dan Z tampil dalam empat kriteria yakni E, O, R dan F.

2.1.2.2 Penilaian Setiap Pilihan untuk Harga *(Expense)*

	X	Y	Z
X	1	5	9
Y	1/5	1	3
Z	1/9	1/3	1

Tabel 2.6: Rating Setiap Mesin Untuk Harga *(expense)*[1]

Ini berarti bahwa X jauh lebih baik daripada Y dalam hal biaya dan terlebih lagi untuk Z. Angka biaya sebenarnya dapat digunakan namun akan mengubah nilai matriks ini, yang relatif terhadap faktor penilaian kualitatif lainnya. Vektor eigen untuk matriks ini adalah (0,751, 0,188, 0,071), sama seperti yang diharapkan, dan *CR* adalah 0,072, sehingga penilaiannya dapat diterima konsisten.

Tiga matriks berikutnya masing-masing merupakan penilaian dari manfaat relatif X, Y dan Z sehubungan dengan kemampuan pengoperasian, keandalan dan fleksibilitas (hanya untuk mengingatkan Anda, X murah dan mudah dioperasikan namun tidak terlalu dapat diandalkan dan tidak dapat dengan mudah disesuaikan dengan penggunaan, lain halnya dengan Y agak lebih mahal, cukup mudah dioperasikan, sangat andal namun tidak terlalu mudah beradaptasi. Akhirnya, Z sangat mahal, tidak mudah dioperasikan, sedikit kurang dapat diandalkan dibanding Y namun diklaim memiliki jangkauan yang luas. Penggunaan alternatif):

2.1.3 Penilaian Setiap Pilihan Untuk Operabilitas

	X	Y	Z
X	1	1	5
Y	1	1	3
Z	1/5	1/3	1

Tabel 2.7: Eigenvector (0.480, 0.406, 0.114)[1]

2.1.4 Penilaian Setiap Pilihan Untuk Keandalan (*Reliability*)

	X	Y	Z
X	1	1/3	1/9
Y	3	1	1/3
Z	9	3	1

Tabel 2.8: Eigenvector (0.077, 0.231, 0.692)[1]

2.1.5 Penilaian Setiap Pilihan Untuk Fleksibilitas

	X	Y	Z
X	1	1/9	1/5
Y	9	1	2
Z	5	1/2	1

Tabel 2.9: Eigenvector (0.066, 0.615, 0.319)[1]

Alasan bahwa nilai Y lebih baik daripada Z pada kriteria ini adalah bahwa perusahaan tersebut tidak benar-benar mempercayai klaim pabrikan untuk Z. AHP menangani pendapat dan firasat semudah fakta.

2.1.6 Tahap akhir adalah membuat matriks vektor eigen untuk X, Y dan Z

	E	O	R	F
X	0.751	0.480	0.077	0.066
Y	0.178	0.406	0.231	0.615
Z	0.071	0.114	0.692	0.319

Tabel 2.10: Eigenvector 0.232, 0.402, 0.061, 0.305)[1]

Matriks ini, yang kita sebut *Option Performance Matriks (OPM)*, merangkum kemampuan masing-masing dari tiga mesin dalam hal apa yang diinginkan perusahaan. Didapatkan hasil X jauh lebih baik daripada Y dan Z dalam hal biaya; X sedikit lebih baik dari Y dalam hal pengoperasian, bagaimanapun, X adalah nilai terbatas dalam hal keandalan dan fleksibilitas. Berbagai pilihan tersebut berhubungan dengan seperangkat kriteria yang dipilih oleh hipotetis ini. Bagi perusahaan lain yang keandalannya lebih penting dan ingin mengurangi biaya, ketiga mesin mungkin bisa mencetak skor yang sangat berbeda.

Langkah terakhir adalah memperhitungkan penilaian perusahaan mengenai kepentingan relatif E, O, R dan F. Bagi perusahaan yang lebih mementingkan fleksibilitas, Y akan ideal. Jika keandalan, maka mesin Z adalah pilihan pertama. Nilai *Relatif Value Vector* adalah: 0.232, 0.402, 0.061, 0.305. Lalu dilakukan perhitungan mengalikan OPM dengan RVV untuk mendapatkan vektor kemampuan masing-masing mesin ini untuk memenuhi kebutuhan perusahaan. Maka didapatkan hasil: 0,392, 0,406, 0,204; dan bisa disebut vektor *Value For Money* (VFM). Dalam matriks aljabar, OPM * RVV = VFM atau:

performance*requirement = value for money.

Tiga angka dalam VFM adalah hasil akhir perhitungan. Pertama, X yang memiliki nilai 0,392 nampaknya sedikit lebih

buruk dalam hal kemampuannya memenuhi kebutuhan perusahaan daripada Y pada 0,406. Z berada jauh di belakang pada 0,204 dan agak buruk dalam memenuhi persyaratan perusahaan dalam kasus ilustratif ini[1].

2.2 *Docker*

Docker[3] adalah sebuah aplikasi yang bersifat open source yang berfungsi sebagai wadah atau container untuk mengepak atau memasukkan sebuah perangkat lunak secara lengkap beserta semua hal lainnya yang dibutuhkan oleh perangkat lunak tersebut agar dapat berfungsi.

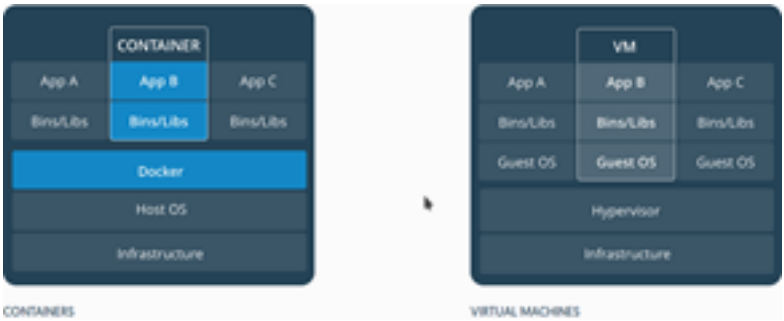
2.2.1 *Docker Images*

Docker *images*[4] adalah sebuah *blueprint* atau dasar dari aplikasi berbasis Docker yang bersifat *read-only*. *Blueprint* ini sebenarnya adalah sebuah sistem operasi atau sistem operasi yang telah dipasang berbagai aplikasi dan pustaka pendukung. Docker *images* berfungsi untuk membuat Docker *container*, yang mana dengan menggunakan satu Docker *image* dapat dibuat banyak Docker *container*. Dengan menggunkaan Docker image, permasalah yang dikenal dengan ”*dependency hell*”, dimana sulitnya untuk melengkapi dependensi sebuah aplikasi, dapat diselesaikan karena semua kebutuhan aplikasi sudah berada di dalamnya.

2.2.2 *Docker Container*

Docker container[4] bisa dikatakan sebagai sebuah folder, dimana docker container ini dibuat dengan menggunakan docker container. Setiap docker container dijalankan maka akan terbentuk layer tepat di atas docker images. Contohnya saat menggunakan image Ubuntu, kemudian membuat sebuah

container dari image Ubuntu tersebut dengan nama “Ubuntuku”. Kemudian lakukan pemasangan sebuah perangkat lunak, misalnya nginx, maka secara otomatis container Ubuntuku akan berada di atas layer image Ubuntu. Docker container ke depannya dapat dibangun sehingga akan menghasilkan sebuah docker images, dan docker images yang dihasilkan dari docker container ini dapat digunakan kembali untuk membuat docker container yang baru.



Gambar 2.1: Visualisasi perbedaan container dengan virtual machine

Pada *virtual machine* mencakup aplikasi, *library* dan *binary*, serta seluruh guest OS, dimana bisa mencapai puluhan GB data. Sedangkan *container* terdapat aplikasi dan semua pendukungnya, tetapi membagi kernel dengan *container* lainnya, menjalankan aplikasi yang terisolasi pada host operating system. *container* Docker tidak terikat dengan infrastruktur tertentu: mereka dapat dijalankan pada komputer manapun, pada infrastuktur apapun, dan di cloud.

2.3 Python

Python[5] adalah bahasa pemrograman interpretatif multiguna dengan prinsip agar sumber kode yang dihasilkan

memiliki tingkat keterbacaan yang baik. Python diklaim sebagai bahasa yang menggabungkan kapabilitas, kemampuan, dengan sintaksis kode yang sangat jelas, dan dilengkapi dengan fungsionalitas pustaka standar yang besar serta komprehensif. Python mendukung beragam paradigma pemrograman, seperti pemrograman berorientasi objek, pemrograman imperatif, dan pemrograman fungsional. Python dapat digunakan untuk berbagai keperluan pengembangan perangkat lunak dan dapat berjalan di berbagai platform sistem operasi.

2.4 Flask

Flask[6] adalah sebuah kerangka kerja web. Artinya, Flask menyediakan perangkat, pustaka, dan teknologi yang memungkinkan seorang pengembang untuk membangun aplikasi berbasis web. Aplikasi web yang bisa dibangun bisa berupa sebuah halaman web, blog, wiki, bahkan untuk web komersial. Flask dibangun berbasiskan pada Werkzeug, Jinja 2, dan MarkupSafe yang mana menggunakan bahasa pemrograman Python sebagai basisnya. Flask sendiri pertama kali dikembangkan pada tahun 2010 dan didistribusikan dengan lisensi BSD.

Flask termasuk sebagai perangkat kerja micro karena tidak membutuhkan banyak perangkat atau pustaka tertentu agar bisa bekerja. Flask tidak menyediakan fungsi untuk melakukan interaksi dengan basis data, tidak mempunya validasi *form* atau fungsi lain yang umumnya bisa digunakan dan disediakan pada sebuah kerangka kerja. Meskipun memiliki kemampuan yang minim, tapi Flask mendukung dan memberikan kemudahan bagi pengembang untuk menambahkan pustaka sendiri untuk mendukung aplikasinya. Berbagai pustaka seperti validasi *form*, mengunggah file, berbagai macam teknologi autentifikasi bisa digunakan dan tersedia untuk Flask. Bahkan pustaka-pustaka

pendukung tersebut diperbarui lebih sering dibandingkan dengan Flasknya sendiri.

2.5 MySQL

MySQL[7] adalah sebuah perangkat lunak terbuka untuk melakukan manajemen basis data SQL atau DBMS. MySQL ditulis dalam bahasa pemrograman C dan C++. MySQL merupakan salah satu perangkat lunak terbuka yang banyak disukai oleh pengembang dan digunakan dalam banyak aplikasi web. Parser SQL yang digunakan ditulis dalam yacc. MySQL bekerja pada banyak platform, seperti AIX, BSDi, FreeBSD, HP-UX, eComStation, i5/OS, IRIX, Linux, macOS, Microsoft Windows, NetBSD, Novell NetWare, OpenBSD, OpenSolaris, OS/2 Warp, QNX, Oracle Solaris, Symbian, SunOS, SCO OpenServer, SCO. MySQL tersedia sebagai perangkat lunak gratis di bawah lesensi *GNU General Public License* (GPL), tetapi juga tersedia lisensi komersial untuk kasus-kasus dimana penggunanya tidak cocok dengan penggunaan GPL.

Setiap pengguna dapat secara bebas menggunakan MySQL, namun dengan batasan perangkat lunak tersebut tidak boleh dijadikan produk turunan yang bersifat komersial. MySQL sebenarnya merupakan turunan salah satu konsep utama dalam basisdata yang telah ada sebelumnya; SQL (Structured Query Language). SQL adalah sebuah konsep pengoperasian basisdata, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis.

Kehandalan suatu sistem basisdata (DBMS) dapat diketahui dari cara kerja pengoptimasi-nya dalam melakukan proses perintah-perintah SQL yang dibuat oleh pengguna maupun program-program aplikasi yang memanfaatkannya. Sebagai peladen basis data, MySQL mendukung operasi basisdata

transaksional maupun operasi basisdata non-transaksional. Pada modus operasi non-transaksional, MySQL dapat dikatakan unggul dalam hal unjuk kerja dibandingkan perangkat lunak peladen basisdata kompetitor lainnya. Namun pada modus non-transaksional tidak ada jaminan atas reliabilitas terhadap data yang tersimpan, karenanya modus non-transaksional hanya cocok untuk jenis aplikasi yang tidak membutuhkan reliabilitas data seperti aplikasi blogging berbasis web (wordpress), CMS, dan sejenisnya. Untuk kebutuhan sistem yang ditujukan untuk bisnis sangat disarankan untuk menggunakan modus basisdata transaksional, hanya saja sebagai konsekuensinya unjuk kerja MySQL pada modus transaksional tidak secepat unjuk kerja pada modus non-transaksional.

2.6 *Multi Criteria Decision Making (MCDM)*

Pengambilan keputusan (*decision making*)[2] adalah proses untuk memilih berbagai alternatif pilihan berdasarkan beberapa kriteria. Di setiap pilihan, terdapat beberapa faktor dan kriteria yang akan dipertimbangkan dan beberapa alternatif pilihan yang harus diputuskan. Pada pengambilan keputusan kelompok kriteria dan alternatif ini harus jelas dan harus ditentukan terlebih dahulu sebelum kita memberikan beberapa nilai penilaian atau nilai evaluasi pada mereka. Penentuan kriteria dan alternatifnya sangat subjektif. Perhatikan bahwa daftar kriteria dan alternatif di atas tidaklah daftar yang lengkap. Pada daftar tidak mencakup semua kriteria yang mungkin atau semua kemungkinan alternatif. Tidak ada kriteria yang benar atau salah karena opini subjektif. Orang yang berbeda dapat menambahkan atau mengurangi daftar tersebut. Beberapa faktor dapat dikombinasikan bersama dan beberapa kriteria dapat dipecah menjadi kriteria yang lebih detail. Sebagian besar keputusan dibuat berdasarkan penilaian individu. Sewaktu kita mencoba

Goal	Criteria	Alternatives
Decide best school	• Distance, • Reputation, • Cost, • Teacher kindliness	Name of schools under consideration
Finding best apartment	• Price, • Down payment, • Distance from shops, • Distance from work/school • Neighbor's Friendliness	List of apartments under consideration
Select best politician	• Charm • Good working program • Benefit for our organization • Attention to our need	List of candidates
Determine thesis topic	• Fast to finish, • Research Cost , • Level of Attractiveness,	List of thesis topics
Buy car	• Initial Price • Operating & Maintenance cost, • Service and comfort, • Status	Car's trade mark (Honda, GM, Ford, Nissan etc.)
Decide whether to buy or to rent a machine	• Total cost (capital, maintenance, operational) • Service • Time to operate • Interconnection with other machines	Rent or Buy

Gambar 2.2: Contoh kriteria dan alternatif beberapa keputusan pembuatan keputusan[2]

membuat keputusan kita se rasional mungkin, kita perlu mengukur opini subyektif ini menjadi nilai subjektif. Nilai adalah angka dalam rentang tertentu; katakanlah dari 1 sampai 10 atau -5 sampai 5. Nilai dapat berupa urutan angka (nomor urut) dan Anda bahkan dapat menentukan rentang yang berbeda untuk setiap faktor. Nilai yang lebih tinggi menunjukkan tingkat faktor atau nilai yang lebih tinggi. Sekarang Anda melihat bahwa tidak hanya kriteria dan alternatifnya bersifat subyektif, bahkan nilainya juga subjektif. Mereka tergantung pada Anda sebagai pengambil keputusan.

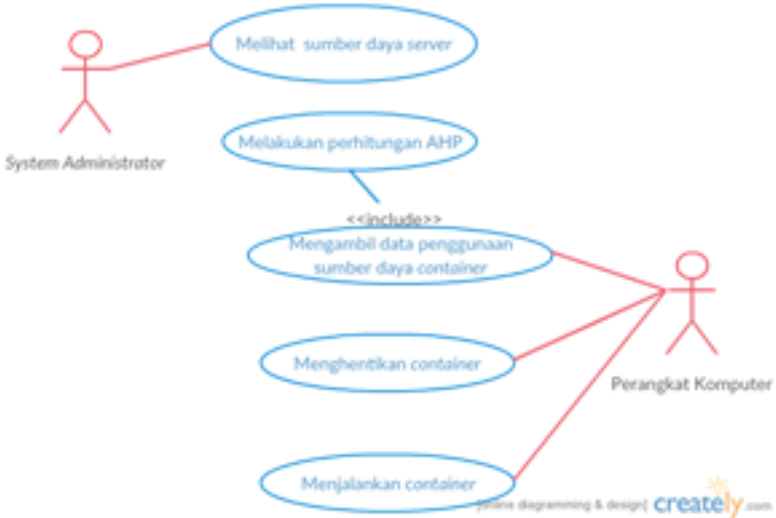
BAB III

DESAIN DAN PERANCANGAN

Pada bab ini dibahas mengenai analisis dan perancangan sistem.

3.1 Kasus Penggunaan

Terdapat dua aktor dalam sistem yaitu sistem administrator dan perangkat komputer dan jaringan. Diagram kasus penggunaan digambarkan pada Gambar 3.1



Gambar 3.1: Diagram Kasus Penggunaan

Diagram kasus penggunaan pada 3.1 dideskripsikan masing-masing pada Tabel 3.1.

Tabel 3.1: Daftar Kode Kasus Penggunaan

Kode Kasus Penggunaan	**Nama Kasus Penggunaan**	**Keterangan**
UC-0001	Melihat sumber daya server.	*System Administrator* dapat memantau penggunaan sumber daya komputer.
UC-0002	Melakukan perhitungan AHP	Perangkat komputer menghitung nilai AHP untuk setiap *container* berdasarkan sumber daya dan waktu akses terakhir setiap *container*.
UC-0003	Mengambil data penggunaan sumber daya *container*	*Middleware* mengambil data penggunaan sumber daya komputer lalu menyimpannya di *database*.
UC-0004	*Stop container*	Perangkat komputer dapat menghentikan *container* yang sedang berjalan berdasarkan hasil perhitungan AHP.
UC-0005	*Start container*	Perangkat komputer dapat menjalankan *container* yang mati jika terdapat permintaan akses.

3.2 Arsitektur Sistem

Pada Sub-bab ini, dibahas mengenai tahap analisis arsitektur, analisis teknologi dan desain sistem yang akan dibangun.

3.2.1 Desain Umum Sistem

Sistem yang akan dibuat adalah sebuah sistem yang dapat menjaga ketersediaan sumber daya seperti CPU dan RAM pada *server* agar dapat berjalan optimal. Pada *server*, terdapat beberapa *container* yang berisi aplikasi Moodle[8], dimana setiap *docker container* berisi satu buah *course* spesifik yang dapat diakses oleh pengguna.

Setiap ada permintaan ke *course* tertentu, permintaan akses akan diterima oleh *middleware*, lalu dialihkan ke *docker container* yang spesifik. Setiap ada permintaan akses ke *course* tertentu, maka pada masing - masing *container* Moodle akan mencatat *last time access* (waktu akses terakhir) yang nantinya akan digunakan oleh *middleware* untuk menghitung tingkat kepentingan masing - masing *container*.

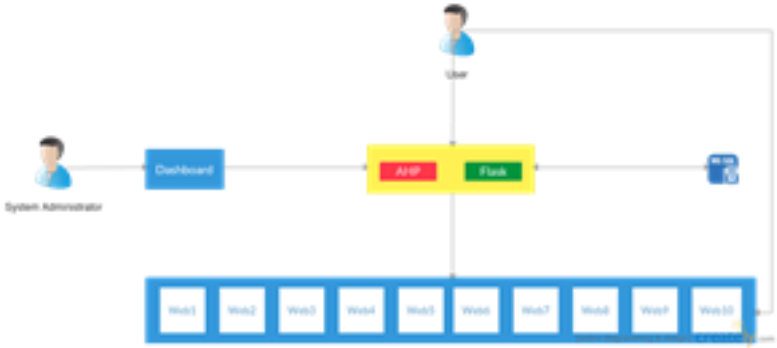
Perhitungan bobot dilakukan untuk menghentikan (*stop*) *container* yang tingkat kepentingannya rendah. Penentuan bobot untuk setiap container dilakukan dalam setiap durasi yang dapat ditentukan nantinya di *middleware*. Perhitungan bobot menggunakan algoritma *Analytical Hierarcy Process*, dengan menggunakan tiga parameter yakni CPU, RAM, dan *last time access*. Secara garis besar, ada tiga bagian penyusun dari aplikasi ini, yakni:

1. Penyimpanan Data Sumber Daya *container* dan Hasil AHP
   Penyimpanan data adalah sebuah lingkungan basis data yang befungsi sebagai penyimpanan terpusat semua log *container* dan hasil perhitungan *AHP*.
2. *Middleware*
   Middleware adalah penghubung antara dasbor dengan

penyimpanan data serta tempat menghitung nilai AHP. Selain itu, *middleware* juga berfungsi sebagai aplikasi untuk penyimpanan data penggunaan sumber daya *server* serta pengambilan data setiap penggunaan sumber daya *container*.

3. Dasbor
   Dasbor adalah halaman yang digunakan sistem administrator untuk menampilkan data penggunaan sumber daya komputer.

Secara visual, desain sistem secara umum digambarkan pada Gambar 3.2. *Middleware* mengambil data penggunaan sumber daya masing - masing *container*, kemudian *middleware* menyimpan data pengunaan sumber daya dari setiap *container* yang diperlukan saja (penggunaan CPU, RAM, *lta*) untuk menghitung nilai AHP. Setelah data pengunaan sumber daya *container* tersimpan, *middleware* akan menghitung nilai AHP dan menentukan *container* mana yang lebih baik untuk di hentikan, lalu menyimpan kembali hasil perhitungan di *database*. Sistem administrator dapat melihat penggunaan sumber daya *server* pada Dasbor.



Gambar 3.2: Desain Umum Sistem

3.2.2 Perancangan Penyimpanan Data Sumber Daya *Container* dan Hasil *AHP*

Penyimpanan data sumber daya *container* dan hasil AHP adalah komponen pada sistem yang berfungsi untuk menyimpan data-data sumber daya setiap *container* yang sudah dikumpulkan oleh *middleware*. Tempat penyimpanan ini akan terpisah dari *middleware* yang ada dan data yang ditempatkan pada penyimpanan ini akan digunakan untuk penghitungan AHP nantinya. Penyimpanan data yang digunakan adalah sebuah basis data SQL (*Structured Query Language*). Nilai-nilai data yang dikumpulkan nantinya akan diidentifikasi berdasarkan waktu data itu didapatkan.

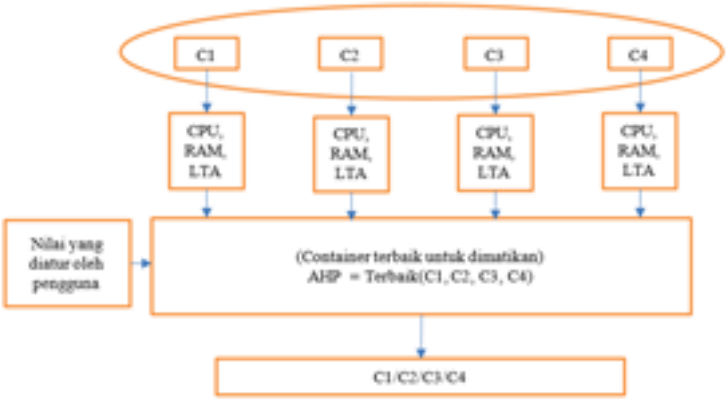
Terdapat beberapa tabel yang akan dibentuk, yakni:

1. Container
   Digunakan untuk menyimpan nama dan sumber daya *container* yang sedang aktif.
2. Result
   Digunakan untuk menyimpan hasil perhitungan AHP.
3. Server-stats
   Digunakan untuk menyimpan data sumber daya penggunaan *server* pada selang waktu tertentu.
4. Stats Digunakan untuk menyimpan data sumber daya penggunaan setiap *container* pada selang waktu tertentu.

3.2.3 Perancangan *Middleware*

Menurut Oxford Dictionaries[9], *middleware* (dalam istilah komputer) adalah suatu perangkat lunak yang menjadi jembatan antara sistem operasi, basis data dan aplikasi dalam sebuah jaringan. Dalam desain umum sistem komponen *middleware* berfungsi sebagai penerima permintaan pengguna untuk mengakses *course*, mengambil data penggunaan sumber daya setiap *container*, dan menentukan *container* terbaik untuk di

hentikan melalui algoritma *AHP*.



Gambar 3.3: Desain *Middleware*

Keterangan:
LTA = Last Time Access (waktu terakhir diakses)
C = Container / RAM

Komponen *middleware* yaitu:

1. *Web Service*
   Web Service berfungsi sebagai penerima permintaan akses ke *course* dari pengguna. Selain itu *web service* juga berfungsi untuk mengolah data dari basis data yang menyimpan sumber daya *container* dan mengembalikan dalam bentuk tipe data *json*, yang nantinya akan diolah oleh dasbor.
   Pada tugas akhir ini, python digunakan sebagai bahasa pemrograman yang digunakan untuk mengimplementasikan komponen *middleware*. Kode python yang digunakan akan dijalankan pada server yang menjalankan *docker container* . Python dipilih karena

mempunyai *support* yang baik dengan *library* docker, dan python juga dapat memudahkan perhitungan AHP karena kaya akan *library scientific*. Lalu, pada bagian penyimpanan data, digunakan MySQL untuk penyimpanan data.

2. Perhitungan *AHP*
   Pada *middleware*, terdapat modul perhitungan AHP yang dijalankan ketika *middleware* telah mengambil data penggunaan sumber daya *server*. Perhitungan AHP menggunakan beberapa parameter sebagai bahan perhitungan yaitu RAM, CPU, dan waktu akhir akses *container*. Pada *middleware*, terdapat file konfigurasi untuk mengatur parameter *judgement* menurut pengguna.

3.2.4 Desain Perhitungan AHP

AHP digunakan untuk menghitung *container* mana yang akan dimatikan. AHP mengembangkan prioritas untuk alternatif yang berbeda dan berdasarkan kriteria yang ada yakni CPU, RAM, dan LTA dan akan diambil sebuah keputusan untuk alternatif tersebut yakni dimatikan. Prioritas awalnya ditetapkan sesuai kepentingan untuk mencapai tujuan, setelah itu prioritas ditetapkan juga untuk performa dari alternatif pada setiap kriteria, prioritas ini ditetapkan berdasarkan penilaian berpasangan dengan menggunakan pengambilan keputusan, atau pengukuran dari suatu skala jika ada[10]. Satu skala umum (diadaptasi dari Saaty) ditunjukkan pada Tabel 3.2

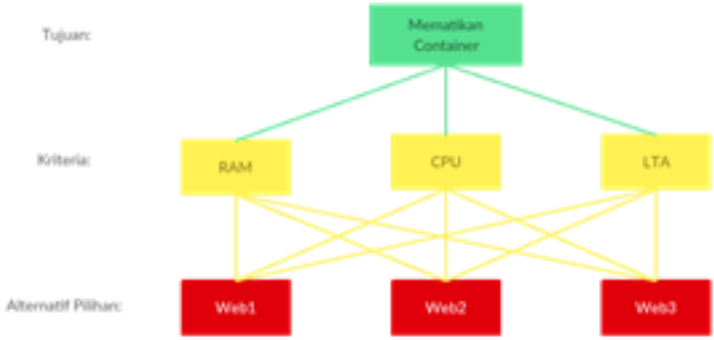
Tabel 3.2: Daftar Skala Prioritas pada AHP

Tingkat Kepentingan	**Definisi**	**Penjelasan**
1	Sama Penting	2 faktor berkontribusi senilai terhadap objektif yang ada.
3	Sedikit Lebih Penting	salah satu faktor lebih penting sedikit dibandingkan faktor yang lain.
5	Lebih Penting	salah satu faktor lebih penting dibandingkan faktor yang lain.
7	Sangat Lebih Penting	salah satu faktor sangat lebih penting dibandingkan faktor yang lain.
9	Benar-benar Lebih Penting	Bukti yang mendukung salah satu faktor dari yang lain telah mencapai kemungkinan yang tertinggi.
2,4,6,8	Nilai Tengah	Nilai saat dimana dibutuhkan kompromi.

Prosedur dasar untuk melaksanakan AHP terdiri dari langkah-langkah berikut:

1. **Penataan permasalahan dan pemilihan kriteria**
   Langkah pertama adalah menguraikan masalah pengambilan keputusan menjadi bagian-bagian penyusunnya. Dalam bentuk yang paling sederhana,

permasalahan terdiri dari tujuan atau fokus permasalahan yaitu menentukan *container* yang akan dimatikan, kriteria (dan subkriteria) pada tingkat menengah yakni RAM, CPU, dan LTA, sedangkan tingkat terendah berisi pilihannya yakni *container1* (Web1) sampai dengan Web *container-n* (Web-n).



Gambar 3.4: Hirarki AHP

2. **Pengaturan prioritas kriteria dengan perbandingan berpasangan (*weighting*)**
   Untuk masing-masing pasangan kriteria, pembuat keputusan diharuskan untuk menjawab pertanyaan seperti ”Seberapa penting CPU terhadap RAM?” Menilai prioritas ”relatif” setiap kriteria dilakukan dengan menetapkan bobot antara 1 Dan 9 seperti pada tabel 3.2 terhadap kriteria yang lebih penting, sedangkan nilai timbal balik dari nilai ini akan diberikan pada pasangan dari kriteria tersebut. Pembobotan ini kemudian akan dinormalisasi untuk mendapatkan bobot untuk setiap kriteria. Nantinya, akan terdapat *file* yang khusus menyimpan pengaturan prioritas kriteria perbandingan pada program.
3. **Perbandingan berpasangan terhadap pilihan pada**

setiap kriteria (*scoring*)
Untuk masing-masing pasangan dalam setiap kriteria, pilihan yang lebih baik akan diberikan nilai pada skala antara 1 dan 9, sementara pilihan lain pasangannya akan diberi nilai sama dengan nilai timbal balik dari nilai ini. Setiap nilai akan menunjukkan seberapa baik pilihan ”x” untuk kriteria ”Y”, misalnya *container* ke 1 dengan CPU. Setelah itu, peringkat akan dinormalisasi.

4. **Mendapatkan skor keseluruhan untuk setiap pilihan**
Pada langkah terakhir, nilai dari setiap pilihan digabungkan dengan bobot kriteria untuk menghasilkan nilai keseluruhan untuk setiap pilihan. Sejauh mana pilihan memenuhi kriteria akan diukur sesuai dengan seberapa penting kriteria tersebut. Hal ini dilakukan dengan rumus:

_(bobot kriteria * bobot OPM)

3.2.5 Desain Dasbor

Dasbor adalah halaman yang digunakan sistem administrator untuk menampilkan data penggunaan sumber daya server. Dasbor adalah aplikasi berbasis web yang dibangun menggunakan VueJS sebagai tampilan depan halaman (frontend). Halaman depan menggunakan Charts.js untuk mendapatkan tampilan yang sederhana dan nyaman digunakan. Dasbor digunakan oleh sistem administrator melihat penggunaan sumber daya komputer berbasis *timeseries*.

Terdapat 2 grafik, yakni penggunaan CPU (atas) dan RAM (bawah). Garis vertikal pada CPU menunjukkan persentase penggunaan dari 0 - 100 % sedangkan pada RAM menunjukkan banyaknya penggunaan RAM dalam satuan Byte dari 0 - maksimum penggunaan RAM. Pada garis horisontal, baik pada gambar CPU maupun RAM menunjukkan baris waktu data, semakin ke kanan maka data semakin terbaru. Panjang garis

horisontal tergantung dari lamanya uji coba dilaksanakan.

Gambar 3.5: Desain Antar Muka Dasbor

(Halaman ini sengaja dikosongkan)

BAB IV

IMPLEMENTASI

Bab ini membahas implementasi sistem secara rinci. Pembahasan dilakukan untuk setiap komponen yang ada yaitu: Penyimpanan data, *docker compose* (untuk membangun *course*), *middleware* dan dasbor.

4.1 Lingkungan Implementasi

Lingkungan pengembangan dilakukan menggunakan Docker Compose dengan spesifikasi *Host* komputer adalah Intel(R) Core(TM) i5-4460S CPU @ 2.90GHz dengan memory 8 GB di Laboratorium Pemrograman I, Teknik Informatika ITS. Perangkat lunak yang digunakan dalam pengembangan adalah sebagai berikut:

- Sistem Operasi Linux Kubuntu 16.04.2 LTS
- Desktop Plasma5
- Editor text Pycharm Community Edition versi 2017.1.2
- Python versi 3.5.2 untuk pengembangan middleware
- git versi 2.7.4 untuk pengolahan versi program
- Docker Compose versi 1.11.2 untuk mengelola Container
- VueJS versi 2 untuk mengelola Dasbor
- Flask Microservice sebagai penyedia web service dan middleware
- Paket LaTeXuntuk pembuatan buku tugas akhir
- Peramban web Google Chrome

4.2 Implementasi Penyimpanan Data

Penyimpanan data dibangun menggunakan basis data MySQL versi 5.7.18.

4.2.1 Menjalankan Database MySQL

Mysql dipasang pada server pengembangan. Untuk memulai pemasangan MySql, terlebih dahulu lakukan proses instalasi MySql seperti pada 1.4. Setelah MySQL dipasang pada *server*, dapat dijalakan dengan perintah `sudo systemctl start mysql`

4.2.2 Skema penyimpanan data log *Docker Container* pada MySQL

Tabel `stats` seperti yang terlihat pada 4.1, digunakan untuk menyimpan data penggunaan sumber daya *container* yang berjalan pada *server*.

Tabel 4.1: Tabel stats

No	Kolom	Tipe	Keterangan
1	id	varchar(50)	Sebagai primary key pada tabel, berupa gabungan id container dengan waktu saat ini dalam satuan detik
2	container_id	varchar(50)	Menyimpan id container yang log nya diambil
3	container_name	varchar(50)	Menyimpan nama container yang log nya diambil
4	cpu	int	Nilai persentase cpu container yang digunakan

5	memory	float	Nilai memory dalam satuan megabyte yang digunakan oleh container
6	memory_percentage	float	Nilai persentase memory container yang digunakan oleh container
7	last_time_access	datetime	Waktu akses web server pada sebuah container terakhir kali
8	last_time_percentage	float	Nilai persentase waktu akses web server pada sebuah container terakhir kali
9	timestamps	datetime	Waktu pada komputer saat data log disimpan

4.2.3 Skema Penyimpanan Data Log *Server* Pada MySQL

Penyimpanan data penggunaan sumber daya *server* disimpan di tabel `server-stats` seperti dalam tabel 4.2,

Tabel 4.2: Tabel server-stats

No	Kolom	Tipe	Keterangan
1	cpu	float	Menyimpan nilai persentase cpu server

2	memory	float	Menyimpan nilai persentase memory server
3	timestamps	datetime	Menyimpan waktu saat log server diambil

4.2.4 Skema Penyimpanan Hasil AHP Pada MySQL

Penyimpanan hasil perhitungan AHP disimpan pada tabel `result` seperti dalam tabel 4.2.

Tabel 4.3: Tabel server-stats

No	Kolom	Tipe	Keterangan
1	container_id _hours	varchar(50)	Menyimpan *id container* yang terpilih berdasarkan penggunaan sumber daya berdasarkan durasi jam.
2	container_id _days	varchar(50)	Menyimpan *id container* yang terpilih berdasarkan penggunaan sumber daya berdasarkan durasi hari.
3	container_id _weeks	varchar(50)	Menyimpan *id container* yang terpilih berdasarkan penggunaan sumber daya berdasarkan durasi mingguan.

4	score_hours	float	Menyimpan nilai skor *container* yang terpilih berdasarkan penggunaan sumber daya berdasarkan durasi jam.
5	score_days	float	Menyimpan nilai skor *container* yang terpilih berdasarkan penggunaan sumber daya berdasarkan durasi hari.
6	score_weeks	float	Menyimpan nilai skor *container* yang terpilih berdasarkan penggunaan sumber daya berdasarkan durasi mingguan.
7	hour_from	datetime	Menyimpan tanggal pengambilan dari durasi jam dimulai.
8	day_from	datetime	Menyimpan tanggal pengambilan dari durasi hari dimulai.
9	week_from	datetime	Menyimpan tanggal pengambilan dari durasi mingguan dimulai.
10	timestamps	datetime	Menyimpan tanggal data disimpan.

4.2.5 Skema Penyimpanan *Container* Yang Aktif

Penyimpanan *container* yang aktif disimpan pada tabel `containers` seperti dalam tabel 4.4.

Tabel 4.4: Tabel containers

No	Kolom	Tipe	Keterangan
1	container_id	varchar(50)	Menyimpan *id container*yang sedang aktif.
2	name	varchar(50)	Menyimpan nama *container* yang sedang aktif.
3	status	varchar(50)	Menyimpan status *container*.
4	timestamps	datetime	Menyimpan tanggal data disimpan.

4.3 Implementasi *Docker Compose*

Docker Compose digunakan untuk membangun paket container yang saling terhubung. Membangun Docker Compose menggunakan *container* MariaDB sebagai database penyimpanan data dan Moodle sebagai *container* yang akan diakses. Digunakan *container* MariaDB dan Moodle yang sudah dikonfigurasi oleh Bitnami[8].

4.3.1 Menjalankan *Docker Compose*

Docker Compose dipasang pada server pengembangan. Untuk langkah instalasi, dapat dilakukan sesuai cara pada 1.2

4.3.2 Pembangunan *Docker Compose* Untuk Moodle

Pada *server*, dibangun 10 Docker Compose, yang terdiri dari sebuah Container Moodle dan sebuah Container MariaDB. Masing - masing Container Moodle berisikan sebuah *course* yang dapat diakses oleh pengguna. Berikut adalah salah satu script Docker Compose:

Kode Sumber IV.1: docker-compose.yml

```
version: '2'

services:
  mariadb:
    container_name: 'mariadb1'
    image: 'bitnami/mariadb:latest'
    environment:
        ALLOW_EMPTY_PASSWORD=yes
    volumes:
        './mariadb_data:/bitnami/mariadb'
  moodle:
    container_name: 'moodle1'
    image: 'bitnami/moodle:latest'
    ports:
        '10001:80'
        '9001:443'
    volumes:
        './moodle_data:/bitnami/moodle'
        './apache_data:/bitnami/apache'
        './php_data:/bitnami/php'
    depends_on:
        mariadb

volumes:
  mariadb_data:
```

```
    driver: local
  moodle_data:
    driver: local
  apache_data:
    driver: local
```

4.3.3 Menjalankan *Docker Compose* Sebagai Layanan Moodle

Dapat dilihat pada kode sumber IV.1, untuk menjalankan Docker Compose tersebut, dapat dilakukan dengan perintah `docker-compose up` pada folder yang sama dengan file docker-compose.yml berada. Cukup membuat 10 folder, beri nama folder tersebut dari 1 sampai 10, lalu letakkan script IV.1 pada setiap folder, dengan perubahan container_name pada setiap folder disesuaikan. Misalnya pada folder 5, maka container_name moodle adalah moodle5 dan mariadb5.

4.4 Implementasi *Middleware*

Middleware dibangun sebagai perantara antara penyimpanan data, dengan dasbor dan sebagai tempat untuk memproses *AHP*. Middleware dibangun menggunakan Bahasa Python dan Microservice Flask. Untuk menjalankan Python, terlebih dahulu paket Python versi 3.5 harus terpasang pada komputer server.

4.4.1 Implementasi Mengalihkan Permintaan *Course* ke *Container* Yang Tepat

Pada *middleware*, jika terdapat permintaan ke *course* tertentu, maka akan langsung diarahkan ke *container* yang tepat. Ketika mengalihkan permintaan, *middleware* akan memeriksa apakah container dalam keadaan siap atau tidak. *Container* dikatakan siap jika *container* dalam keadaan *running*(berjalan).

Untuk memastikan *container* yang diakses dalam keadaan *running*, maka setiap permintaan akses akan memanggil fungsi `start_docker` untuk menjalankan *container* jika dalam keadaan mati.

Kode Sumber IV.2: mengalihkan permintaan *course* ke *container* yang tepat

```
@application.route("/moodle/<moodle_id>")
def moodle(moodle_id):
    moodle = 'moodle' + moodle_id
    utils.start_docker(moodle)
    if moodle_id == "10":
        moodle_id = str(10010)
    else:
        moodle_id = str(1000) + moodle_id

    url = "http://localhost:" + moodle_id
    while (requests.head(url).status_code
       !=200):
        time.sleep(1)

    return redirect(url, code=302)
```

Dapat dilihat pada kode sumber IV.2 ketika terdapat permintaan akses ke `/moodle/1` misalnya, maka *middleware* akan memanggil fungsi `utils.start(moodle)` yang dapat dilihat pada kode sumber IV.3

Kode Sumber IV.3: mengalihkan permintaan *course* ke *container* yang tepat

```
def start_docker(id_or_name):
    con = client.containers.get(id_or_name)
    number = re.search(r'\d+', con.name).
       group()
    maridb = client.containers.get("mariadb
       " + number)
```

```
    if con.status == 'exited':
        con.start()
        maridb.start()
```

Middleware akan mengecek status dari *container* yang dituju. Jika status *container* sedang *exited* maka middleware akan membangkitkannya.

4.4.2 Implementasi Pengambilan Data Penggunaan *Server*

Pada *middleware*, disediakan *route* untuk mengambil data penggunaan sumber daya server. Ketika *route* tersebut diakses, maka *middleware* akan merespon dengan memberikan data penggunaan server secara *timeseries* berformat json, seperti pada kode sumber IV.4.

Kode Sumber IV.4: mengalihkan permintaan *course* ke *container* yang tepat

```
@application.route("/server stats", methods
   =['GET', 'POST'])
def server_stats():
    table_name = "server_stats"
    if request.method == 'GET':
        res = utils.log(table_name)
        return jsonify(res)
    if request.method == 'POST':
        limit = True
        res = utils.log(table_name, limit,
           request.form["from"], request.
           form["to"])
        return jsonify(res)
```

Nantinya, dasbor akan memanggil *route* tersebut dan diolah dalam bentuk grafik untuk memudahkan *System Administrator* dalam melihat penggunaan sumber daya *server*.

4.4.3 Implementasi *Background Scheduler* Untuk Menjalankan *Middleware* pada *Background Process*

Pada *middleware*, terdapat sebuah fungsi untuk mengambil data penggunaan sumber daya masing - masing *container* dan *server* serta menghitung nilai AHP secara *background process* agar tidak mengganggu layanan web service Flask.

Kode Sumber IV.5: mengalihkan permintaan *course* ke *container* yang tepat

```
scheduler = BackgroundScheduler()
scheduler.start()
scheduler.add_job(
    func=utils.stats,
    trigger=IntervalTrigger(minutes=10),
    id='get_docker_stats',
    name='Get Docker Stats every %s minutes
        ' % get_stats,
    replace_existing=True)
# Shut down the scheduler when exiting the
   app
atexit.register(lambda: scheduler.shutdown
   ())
```

Dapat dilihat pada kode sumber IV.5, *middleware* memanggil fungsi `utils.stats` yang akan mengambil data *container*, server , dan menghitung nilai AHP. Pemanggilan fungsi tersebut dilakukan dengan interval waktu setiap 10 menit (tergantung referensi *system administrator* yang diatur pada file config.ini seperti pada B.4).

4.4.4 Implementasi Pengambilan Data *Last Time Access* Pada Container Moodle

Pengambilan data *last time access* pada setiap moodle (10 buah) dilakukan dengan cara mengambil log dari Docker

Compose setiap moodle lalu mengambil *timestamp* terakhir yang statusnya ’200’ (sukses). Berikut adalah code dari pengambilan data *last time access*.

Kode Sumber IV.6: Mengambil Data LTA

```
def get_LTA_Data(con):
    conName = con.name
    timepercentage = 0.0
    date_last_access = datetime.datetime.
       min

    # Check if the container is running
    if (con.status != 'running'):
        raise ValueError('"%s" container is
            not running' % conName)

    now = datetime.datetime.now()
    con_log_all = con.logs(stream=False,
       timestamps=1)
    if b'200' in con_log_all:
        last_hit_start = int(str(
           con_log_all).rfind('[')) + 1
        last_hit_end = int(str(con_log_all)
           .rfind(']'))

        date_last_access = str(con_log_all)
           [last_hit_start:last_hit_end]
        date_last_access = datetime.
           datetime.strptime(
           date_last_access, '%d/%b/%Y:%H:%
           M:%S %z')
        date_last_access = utc_to_local(
           date_last_access)
```

```
        date_now = datetime.datetime.utcnow
            ()
        date_now.replace(tzinfo=pytz.utc).
            astimezone(local_tz)
        date_now = utc_to_local(date_now)
        difference_date = date_now
            date_last_access
        difference_date = difference_date.
            total_seconds()
        day = 86400  # seconds

        timepercentage = difference_date /
            day * 100
    else:
        con_log = "No Data"

    if timepercentage is not 0:
        return timepercentage,
            date_last_access.replace(tzinfo=
            None)
    else:
        return con_log
```

Fungsi diatas akan membaca log dari *docker compose* pada setiap *container* dan mengambil data *timestamp container* tersebut untuk nantinya disimpan pada database. Untuk perhitungan AHP, karena RAM dan CPU dalam satuan persen, maka variabel *last time access* juga harus disamakan(dijadikan persentase). untuk itu dilakukan prosen konversi ke satuan persen, dimana 100 persen merupakan satu hari (24 jam).

4.4.5 Implementasi Pengambilan Data *Memory* Pada Container Moodle

Kode Sumber IV.7: Mengambil Data RAM

```
conName = con.name
    memorypercentage = 0.0

    # Check if the container is running
    if (con.status != 'running'):
        raise ValueError('"%s" container is
            not running' % conName)

    # Get Memory Usage in percentage
    constat = con.stats(stream=False)
    usage = constat['memory_stats']['usage
        ']
    limit = constat['memory_stats']['limit
        ']
    usage_mb = usage / (1024 * 1024)
    limit_mb = limit / (1024 * 1024)
    memorypercentage = usage_mb / limit_mb
        * 100
    return memorypercentage, usage_mb
```

Fungsi diatas akan mengambil data penggunaan sumber data *container* dan menyimpan data penggunaan memory dalam satuan megabytes dan dalam satuan persentase.

4.4.6 Implementasi Pengambilan Data *CPU* Pada *Container* Moodle

Kode Sumber IV.8: Mengambil Data CPU

```
def get_CPU_Percentage(con):
    conName = con.name
    cpupercentage = 0.0
```

```
# Check if the container is running
if (con.status != 'running'):
    raise ValueError('"%s" container is
        not running' % conName)

# Get CPU Usage in percentage
constat = con.stats(stream=False)
prestats = constat['precpu_stats']
cpustats = constat['cpu_stats']

prestats_totalusage = prestats['
   cpu_usage']['total_usage']
stats_totalusage = cpustats['cpu_usage
   ']['total_usage']
numOfCPUCore = len(cpustats['cpu_usage
   ']['percpu_usage'])

prestats_syscpu = prestats['
   system_cpu_usage']
stats_syscpu = cpustats['
   system_cpu_usage']
logging.debug('prestats_syscpu: %s,
   stats_syscpu: %s' % (prestats_syscpu
   , stats_syscpu))

cpuDelta = stats_totalusage
   prestats_totalusage
systemDelta = stats_syscpu
   prestats_syscpu

if cpuDelta > 0 and systemDelta > 0:
    cpupercentage = (cpuDelta /
       systemDelta) * numOfCPUCore
```

```
    formattedcpupert = '{:.1%}'.format(
        cpupercentage)
    logging.debug('cpuDelta: %s,
        systemDelta: %s, cpu: %s' % (
        cpuDelta, systemDelta, cpupercentage
        ))

    logging.info('"%s" Container CPU: %s '
        % (conName, formattedcpupert))

    return (cpupercentage * 100)
```

Pada kode sumber IV.8, dilakukan pengambilan data sebanyak 2x, yakni data saat ini dan data sebelumnya. Itu dilakukan untuk mencari selisih antara 2 buah nilai, dan nilai selisih itulah persentase penggunaan CPU *container*.

4.4.7 Implementasi Perhitungan AHP

Pada *database*, didapatkan data penggunaan sumber daya setiap *container* seperti ditunjukkan pada tabel 4.5 yang diambil pada *server* oleh *middleware*. Nantinya, perhitungan AHP berdasarkan pada penggunaan sumber daya tiap *container* yang aktif. Pada tabel 4.5, terdapat 4 kolom yakni `container_id` yang menampilkan id dari *container*, `cpu_perc` yang menampilkan penggunaan CPU dalam persentase, `memory_perc` yang menampilkan penggunaan RAM dalam persentase, `lta_perc` yang menampilkan penggunaan LTA (waktu akses terakhir) dalam persentase dimana dalam 1 hari (24 jam) merupakan 100 %. Terdapat 10 *container* yang dijalankan pada *server*.

Tabel 4.5: Tabel stats di *database*

No	container_id	cpu perc	memory perc	lta perc
1	7a678c75b4	0	2.35138	0.0586245
2	7180d02ce7	4	2.35866	0.00412776
3	83ed069452	0	1.94498	25.1042
4	3e7b1e779a	0	1.40134	24.9948
5	0b1eaf3825	0	1.76998	25.0847
6	3164d05a95	0	1.64676	25.0867
7	de9c31870c	0	1.53382	25.0689
8	5531fea512	0	1.40134	24.9948
9	5b19e3ac22	0	1.50888	25.0087
10	81ce4b531f	0	1.4676	24.9809

4.4.7.1 Membuat *Comparison Matrix*

Langkah yang pertama adalah menentukan tingkat kepentingan CPU, RAM, dan LTA pada file `config.ini` seperti pada B.4. Pada file `config.ini`, terdapat *header* `[parameter]` dan *header* `[comparison]` dimana potongan kode seperti pada IV.9. Untuk menghasilkan bobot masing - masing kriteria, pengguna cukup mengganti nilai [comparison] sesuai keinginan. Dalam potongan file config.ini B.4, Memory/LTA bernilai 2. Ini berarti, memory dibandingkan dengan *LTA*(last time access/waktu terakhir diakses) agak lebih diprioritaskan dalam penilaian *AHP*. *Range* untuk *comparison matrix* antara 1 sampai dengan 9, seperti yang sudah dijelaskan pada Bab 2.

Kode Sumber IV.9: Tingkat Kepentingan

```
[parameter]
Param1: Memory
Param2: LTA
Param3: CPU
```

```
[comparison]
Memory/LTA: 2
Memory/CPU: 4
LTA/CPU: 2
```

Berdasarkan pengaturan pada file *config.ini* tersebut, maka tersusun *comparison matrix* seperti pada tabel 4.6.

Tabel 4.6: *Comparison Matrix*

	Memory	**LTA**	**CPU**
Memory	1	0.5	0.25
LTA	2	1	0.5
CPU	4	2	1

4.4.7.2 Menghitung Bobot Untuk Parameter

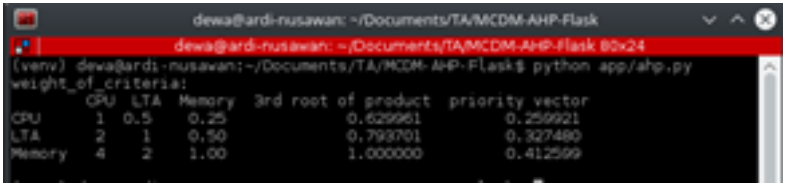
Proses perhitungan *AHP* yang pertama adalah menghitung *weight of criteria*. *Weight of criteria* (bobot masing - masing kriteria) adalah nilai bobot setiap kriteria dibandingkan dengan kriteria lainnya. Terdapat tiga langkah untuk menghitung *Weight of Criteria*, yakni:

1. Hitung 3^{rd} *root of product*
2. *Priority Vector* = nilai dari 3^{rd} *root of product* / jumlah dari 3^{rd} *root of product*
3. Hitung jumlah nilai dari setiap kolom

Tabel 4.7: Pengaturan Prioritas Kriteria Dengan Perbandingan Berpasangan

	Memory	**LTA**	**CPU**	***3rd Root of Product***	***Priority Vector***
CPU	1	0.5	0.25	0.629961	0.259921
LTA	2	1	0.5	0.793701	0.327480

Memory	4	2	1	1.000000	0.412599
Jumlah	7	3.5	1.75	2.638	1.000000



Gambar 4.1: Pengaturan Prioritas Kriteria Dengan Perbandingan Berpasangan

Matriks ini, yang kita sebut *Option Performance Matrix (OPM)*, merangkum kemampuan masing-masing dari tiga parameter apa yang paling penting bagi *server*.

Nantinya, perhitungan *rating* untuk setiap parameter menggukanan cara yang sama. Untuk potongan kodenya, dapat dilihat pada B.1.

4.4.7.2.1 *Rating* Setiap *Container* Untuk *Memory* (RAM)

Perhitungan *rating* dari setiap *container* untuk RAM sama seperti pada perhitungan menentukan bobot untuk setiap parameter yang langkah - langkahnya yakni:

1. Hitung *3rd root of product*
2. *Priority Vector* = nilai dari *3rd root of product* / jumlah dari *3rd root of product*
3. Hitung jumlah nilai dari setiap kolom

Tabel 4.8: *Rating* Setiap *Container* Untuk *Memory* (RAM)

	C1	**C2**	**C3**	**C4**	**C5**
C1	1.000000	1.000000	1.500000	3.000000	2.25
C2	1.000000	1.000000	1.500000	3.000000	2.25

C3	0.666667	0.666667	1.000000	2.000000	1.50
C4	0.333333	0.333333	0.500000	1.000000	0.75
C5	0.444444	0.444444	0.666667	1.333333	1.00
C6	0.333333	0.333333	0.500000	1.000000	0.75
C7	0.222222	0.222222	0.333333	0.666667	0.50
C8	0.111111	0.111111	0.166667	0.333333	0.25
C9	0.222222	0.222222	0.333333	0.666667	0.50
C10	0.111111	0.111111	0.166667	0.333333	0.25
Jumlah	2.111033	4.44443	6.5666634	13.1111	10

Tabel 4.9: *Rating* Setiap *Container* Untuk *Memory* (RAM)

	C6	**C7**	**C8**	**C9**	**C10**
C1	3.000000	4.5	9.0	4.5	9.0
C2	3.000000	4.5	9.0	4.5	9.0
C3	2.000000	3.0	6.0	3.0	6.0
C4	1.000000	1.5	3.0	1.5	3.0
C5	1.333333	2.0	4.0	2.0	4.0
C6	1.000000	1.5	3.0	1.5	3.0
C7	0.666667	1	1.0	2.0	1.045114
C8	0.333333	1.0	1.0	0.5	1.0
C9	0.666667	0.5	2.0	1.0	2.0
C10	0.333333	1.0	1.0	0.5	1.0
Jumlah	114.8	20.5	39	21	40.045

Tabel 4.10: *Rating* Setiap *Container* Untuk *Memory* (RAM)

	3rd Root of Product	***Priority Vector***
C1	2.949461	0.225
C2	2.949461	0.225

C3	1.966307	0.150
C4	0.983154	0.075
C5	1.310871	0.100
C6	0.983154	0.075
C7	0.655436	0.050
C8	0.327718	0.025
C9	0.655436	0.050
C10	0.327718	0.025
Jumlah	12.78	1

4.4.7.2.2 *Rating* Setiap *Container* Untuk CPU

Perhitungan *rating* dari setiap *container* untuk CPU sama seperti pada perhitungan menentukan bobot untuk setiap parameter yang langkah - langkahnya yakni:

1. Hitung *3rd root of product*
2. *Priority Vector* = nilai dari *3rd root of product* / jumlah dari *3rd root of product*
3. Hitung jumlah nilai dari setiap kolom

Tabel 4.11: *Rating* Setiap *Container* Untuk CPU

	C1	**C2**	**C3**	**C4**	**C5**
C1	1.0	0.111111	1.0	1.0	1.0
C2	9.0	1.000000	9.0	9.0	9.0
C3	1.0	0.111111	1.0	1.0	1.0
C4	1.0	0.111111	1.0	1.0	1.0
C5	1.0	0.111111	1.0	1.0	1.0
C6	1.0	0.111111	1.0	1.0	1.0
C7	1.0	0.111111	1.0	1.0	1.0
C8	1.0	0.111111	1.0	1.0	1.0
C9	1.0	0.111111	1.0	1.0	1.0

C10	1.0	0.111111	1.0	1.0	1.0
Jumlah	18	1.9888	18	18	18

Tabel 4.12: *Rating* Setiap *Container* Untuk CPU

	C6	**C7**	**C8**	**C9**	**C10**
C1	1.0	1.0	1.0	1.0	1.0
C2	9.0	9.0	9.0	9.0	9.0
C3	1.0	1.0	1.0	1.0	1.0
C4	1.0	1.0	1.0	1.0	1.0
C5	1.0	1.0	1.0	1.0	1.0
C6	1.0	1.0	1.0	1.0	1.0
C7	1.0	1.0	1.0	1.0	1.0
C8	1.0	1.0	1.0	1.0	1.0
C9	1.0	1.0	1.0	1.0	1.0
C10	18	18	18	18	18

Tabel 4.13: *Rating* Setiap *Container* Untuk CPU

	3rd Root of Product	***Priority Vector***
C1	0.802742	0.055556
C2	7.224674	0.500000
C3	0.802742	0.055556
C4	0.802742	0.055556
C5	0.802742	0.055556
C6	0.802742	0.055556
C7	0.802742	0.055556
C8	0.802742	0.055556
C9	0.802742	0.055556
C10	0.802742	0.055556

Jumlah	14.449	1

4.4.7.2.3 *Rating* Setiap *Container* Untuk LTA

Perhitungan *rating* dari setiap *container* untuk LTA sama seperti pada perhitungan menentukan bobot untuk setiap parameter yang langkah - langkahnya yakni:

1. Hitung *3rd root of product*
2. *Priority Vector* = nilai dari *3rd root of product* / jumlah dari *3rd root of product*
3. Hitung jumlah nilai dari setiap kolom

Tabel 4.14: *Rating* Setiap *Container* Untuk LTA

	C1	**C2**	**C3**	**C4**	**C5**
C1	1.0	1.0	0.111111	0.111111	0.111111
C2	1.0	1.0	0.111111	0.111111	0.111111
C3	9.0	9.0	1.000000	1.000000	1.000000
C4	9.0	9.0	1.000000	1.000000	1.000000
C5	9.0	9.0	1.000000	1.000000	1.000000
C6	9.0	9.0	1.000000	1.000000	1.000000
C7	9.0	9.0	1.000000	1.000000	1.000000
C8	9.0	9.0	1.000000	1.000000	1.000000
C9	9.0	9.0	1.000000	1.000000	1.000000
C10	9.0	9.0	1.000000	1.000000	1.000000
Jumlah	74	74	8.222222	8.222222	8.222222

Tabel 4.15: *Rating* Setiap *Container* Untuk LTA

	C6	**C7**	**C8**	**C9**	**C10**
C1	0.111111	0.111111	0.111111	0.111111	0.111111
C2	0.111111	0.111111	0.111111	0.111111	0.111111

C3	1.000000	1.000000	1.000000	1.000000	1.000000
C4	1.000000	1.000000	1.000000	1.000000	1.000000
C5	1.000000	1.000000	1.000000	1.000000	1.000000
C6	1.000000	1.000000	1.000000	1.000000	1.000000
C7	1.000000	1.000000	1.000000	1.000000	1.000000
C8	1.000000	1.000000	1.000000	1.000000	1.000000
C9	1.000000	1.000000	1.000000	1.000000	1.000000
C10	1.000000	1.000000	1.000000	1.000000	1.000000
Jumlah	8.222222	8.222222	8.222222	8.222222	8.222222

Tabel 4.16: *Rating* Setiap *Container* Untuk LTA

	3rd Root of Product	***Priority Vector***
C1	0.172427	0.013514
C2	0.172427	0.013514
C3	1.551846	0.121622
C4	1.551846	0.121622
C5	1.551846	0.121622
C6	1.551846	0.121622
C7	1.551846	0.121622
C8	1.551846	0.121622
C9	1.551846	0.121622
C10	1.551846	0.121622
Jumlah	12.759622	1

Untuk kode sumber, dilampirkan di B.2.

4.4.7.3 Mendapatkan Skor Keseluruhan Untuk Setiap Pilihan

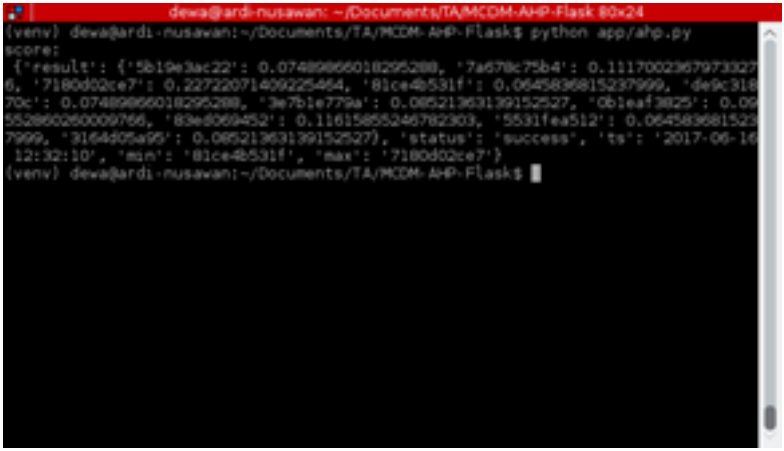
Pada tahap akhir, *middleware* menghitung skor untuk setiap *container* tergantung dari setiap parameter dengan rumus:

_(bobot kriteria * bobot OPM)

Container dengan skor tertinggi adalah pilihan terbaik untuk dihentikan.

Tabel 4.17: Skor Akhir

Kriteria	***Memory***	**CPU**	**LTA**	**Skor**
OPM	0.412599	0.259921	0.327480	1.00000
C1	0.225	0.055556	0.013514	0.11170024
C2	0.255	0.500000	0.013514	0.06458368
C3	0.150	0.555556	0.121622	0.22722071
C4	0.075	0.555556	0.121622	0.11615855
C5	0.100	0.555556	0.121622	0.08521363
C6	0.075	0.555556	0.121622	0.0955286
C7	0.050	0.555556	0.121622	0.08521363
C8	0.025	0.555556	0.121622	0.07489866
C9	0.050	0.555556	0.121622	0.06458368
C10	0.025	0.555556	0.121622	0.07489866
Jumlah	1	1	1	1



Gambar 4.2: *Snapshot* Pemilihan Keputusan dari AHP

Seperti yang terlihat pada gambar 4.2, setelah *middleware* menghitung nilai skor setiap *container*, *middleware* mencari skor tertinggi, lalu mencari *container_id* yang memiliki skor tersebut dan mematikan *container* tersebut. Untuk kasus ini, *container_id* `7180d02ce7` adalah skor tertinggi yang dihasilkan dari hasil perhitungan AHP dan proses *middleware*. Untuk kode sumber dilampirkan di B.3.

4.5 Implementasi Dasbor

Dasbor diimplementasikan menggunakan VueJS. Menu ini dapat melihat penggunaan sumber daya server berbasis urutan waktu. Terdapat 2 grafik, yakni penggunaan CPU (atas) dan RAM (bawah). Garis vertikal pada CPU menunjukkan persentase penggunaan dari 0 - 100 % sedangkan pada RAM menunjukkan banyaknya penggunaan RAM dalam satuan Byte dari 0 - maksimum penggunaan RAM. Pada garis horisontal, baik pada gambar CPU maupun RAM menunjukkan baris waktu data, semakin ke kanan maka data semakin terbaru. Panjang

garis horisontal tergantung dari lamanya uji coba dilaksanakan. Seperti yang ditunjukkan pada gambar 4.3

Gambar 4.3: Dasbor *Penggunaan Resource Server*

(Halaman ini sengaja dikosongkan)

BAB V

PENGUJIAN DAN EVALUASI

5.1 Lingkungan Uji Coba

Pengujian dilakukan pada sebuah komputer dengan spesifikasi:

- Perangkat Keras
  - Processor Intel(R) Core(TM) i3 CPU 550 @3.20GHz
  - RAM 7971200 kB
  - Ethernet interface Express Gigabit Ethernet Controller Speed=100Mbit/s
- Perangkat Lunak
  - Sistem Operasi Kubuntu 16.04.2 LTS
  - Python 3.5
  - Docker
  - Docker Compose
  - Apache JMeter

5.2 Skenario Uji Coba

Uji Coba ini dilakukan untuk menguji apakah fungsionalitas yang diidentifikasikan pada tahap kebutuhan benar-benar diimplementasikan dan bekerja seperti yang seharusnya. Uji coba akan didasarkan pada beberapa skenario untuk menguji kesesuaian respon sistem.

5.2.1 Skenario Uji Fungsionalitas

Uji coba fungsionalitas dilakukan dengan cara menjalankan sistem yang telah dibuat, dan melakukan pengujian terhadap fitur yang telah dibuat. Uji coba fungsionalitas akan berfungsi untuk memastikan sistem sudah memenuhi kebutuhan yang tertera pada Bab 3, yaitu meliputi:

1. Pengujian *System Administrator* dapat melihat sumber daya *server* yang digunakan.

2. Pengujian *middleware* dapat mengambil data penggunaan sumber daya *container*.
3. Pengujian perangkat komputer dapat melakukan perhitungan AHP untuk menentukan *container* mana yang akan dimatikan.
4. Pengujian perangkat komputer dapat menghentikan *container*.

5.2.1.1 Uji Coba *System Administrator* dapat melihat sumber daya server yang digunakan

Aplikasi dasbor digunakan untuk menampilkan *resource server* dan menampilkan hasil AHP. Pada aplikasi dasbor, terdapat sebuah menu untuk melihat data penggunaan CPU dan memory *server*. Rancangan pengujian dan hasil yang diharapkan ditunjukkan dengan tabel 5.1:

Tabel 5.1: Skenario Uji Fungsionalitas Aplikasi Dasbor

No	Menu	Uji Coba	Hasil Harapan
1	Lihat statistik penggunaan sumber daya	*System Administrator* melihat Penggunaan CPU dan Memory *server*	Dasbor dapat menampilkan penggunaan sumber daya yang dipakai oleh *server* secara *timeseries*

5.2.1.2 Uji Coba Mendapatkan Data *Docker Container*

Pengambilan data *container* dilakukan pada perangkat *server* yang menjalankan Moodle. Pada *server*, dijalankan file *Docker Compose* yang berisi *container* Moodle. Terdapat 10 file *docker compose* diletakkan pada folder yang berbeda, dimana setiap *file* merupakan paket Moodle yaitu *container* MariaDB sebagai

database dan *container* Apache2 sebagai web server. Masing - masing *port* Apache2 di *expose* agar dapat diakses oleh pengguna dimulai dari *port* 10001 sampai dengan 10010, yang ketika diakses akan menampilkan website Moodle. *File* konfigurasi *docker compose* dapat dilihat pada tabel IV.1.

Uji coba ini dilakukan dengan menjalankan *middleware* pada *server*. Ketika 10 Moodle telah berjalan, file config.ini seperti ditampilkan padaB.4, pada *middleware* di atur untuk mengambil data setiap 10 menit sekali. Setiap 10 menit sekali, *middleware* akan mengambil data CPU, RAM, dan waktu terakhir diakses(*last time access*, untuk selanjutnya *LTA*) dari setiap *container* Apache2 Moodle dan mengambil data CPU serta RAM pada perangkat *server*.

Rancangan pengujian dan hasil yang diharapkan ditunjukan dengan tabel 5.2:

Tabel 5.2: Skenario Uji Fungsionalitas Mendapatkan Data *Docker Container*

No	**Moodle Container**	**Uji Coba**	**Hasil Harapan**
1	Moodle Container 1	Middleware mengambil data sumber daya *server* dan menyimpannya ke MySQL.	Data *CPU*, RAM, dan *LTA* berhasil disimpan pada MySQL.
2	Moodle Container 2	Middleware mengambil data sumber daya *server* dan menyimpannya ke MySQL.	Data *CPU*, RAM, dan *LTA* berhasil disimpan pada MySQL.

Tabel 5.2: Skenario Uji Fungsionalitas Mendapatkan Data *Docker Container*

No	Moodle Container	Uji Coba	Hasil Harapan
3	Moodle Container 3	Middleware mengambil data sumber daya *server* dan menyimpannya ke MySQL.	Data *CPU*, RAM, dan *LTA* berhasil disimpan pada MySQL.
4	Moodle Container 4	Middleware mengambil data sumber daya *server* dan menyimpannya ke MySQL.	Data *CPU*, RAM, dan *LTA* berhasil disimpan pada MySQL.
5	Moodle Container 5	Middleware mengambil data sumber daya *server* dan menyimpannya ke MySQL.	Data *CPU*, RAM, dan *LTA* berhasil disimpan pada MySQL.
6	Moodle Container 6	Middleware mengambil data sumber daya *server* dan menyimpannya ke MySQL.	Data *CPU*, RAM, dan *LTA* berhasil disimpan pada MySQL.
7	Moodle Container 7	Middleware mengambil data sumber daya *server* dan menyimpannya ke MySQL.	Data *CPU*, RAM, dan *LTA* berhasil disimpan pada MySQL.

Tabel 5.2: Skenario Uji Fungsionalitas Mendapatkan Data *Docker Container*

No	Moodle Container	Uji Coba	Hasil Harapan
8	Moodle Container 8	Middleware mengambil data sumber daya *server* dan menyimpannya ke MySQL.	Data *CPU*, RAM, dan *LTA* berhasil disimpan pada MySQL.
9	Moodle Container 9	Middleware mengambil data sumber daya *server* dan menyimpannya ke MySQL.	Data *CPU*, RAM, dan *LTA* berhasil disimpan pada MySQL.
10	Moodle Container 10	Middleware mengambil data sumber daya *server* dan menyimpannya ke MySQL.	Data *CPU*, RAM, dan *LTA* berhasil disimpan pada MySQL.

5.2.1.3 Uji Coba Sistem Dapat Melakukan Perhitungan AHP

Ketika *middleware* telah menyimpan data penggunaan sumber daya *server* dan data penggunaa sumber daya setiap *container* Moodle, selanjutnya *middleware* menghitung nilai AHP dari setiap *container* yang sedang berjalan. Perhitungan dilakukan pada file `ahp.py` fungsi scoreB.3 yang dipanggil oleh file `utils.py` fungsi `stats` yang dapat dilihat pada kode sumber B.5. Sistem dapat melakukan perhitungan AHP dijelaskan pada tabel 5.3

Tabel 5.3: Skenario Uji Coba Sistem Dapat Melakukan Perhitungan AHP

No	Fungsi	Uji Coba	Hasil Harapan
1	stats()	Memanggil fungsi `score()` pada file `ahp.py`	*Middleware* dapat memanggil fungsi `score()` pada file `ahp.py`
2	score()	Menghitung nilai AHP untuk setiap *container*	*Middleware* berhasil menghitung AHP dan menghasilkan *container* mana yang terpilih untuk dimatikan.

5.2.1.4 Uji Coba Perangkat Komputer Dapat Menghentikan *Container*

Ketika *middleware* telah selesai mengambil data, selanjutnya *middleware* menggunakan nilai AHP yang tertinggi dan mematikan *container* yang terpilih. *Container* dimatikan dengan memanggil *object stop()*.

Tabel 5.4: Mematikan *Container*

No	Fitur	Uji Coba	Hasil Harapan
1	Menghentikan *container*	*Middleware* menghentikan *container*	*Middleware* dapat menghentikan *container* yang terpilih untuk dimatikan berdasarkan perhitungan AHP

5.2.2 Skenario Uji Performa

Pengujian dilakukan dengan menjalankan *middleware* selama rentang waktu 2 jam, 6 jam, dan 1 hari. Selama waktu tersebut, *middleware* mengambil penggunaan sumber daya *server*, penggunaan sumber daya masing - masing *container*, serta menghitung nilai AHP setiap 10 menit, tergantung pengaturan pada file `config.ini`. Uji coba performa akan menguji apakah penggunaan sumber daya (CPU, RAM) *server* dapat lebih optimal jika menggunakan *middleware* atau tidak.

5.3 Hasil Uji Coba dan Evaluasi

Berikut dijelaskan hasil uji coba dan evaluasi berdasarkan skenario yang sudah dijelaskan pada bab 5.2.

5.3.1 Uji Fungsionalitas

Berikut dijelaskan hasil pengujian fungsionalitas pada sistem yang sudah dibangun

5.3.1.1 Uji Coba *System Administrator* Dapat Melihat Penggunaan Sumber Daya *Server*

Uji coba ini dilakukan dengan mengakses alamat ip address `localhost:8080/server-stats`. VueJS digunakan sebagai *server* dasbor. Ketika mengakses alamat tersebut, VueJS akan mengambil data penggunaan sumber daya di *database*, mengolahnya menjadi bentuk `json`, lalu mengirimkan balik ke VueJS. Hasil uji coba seperti tertera pada tabel 5.5

Tabel 5.5: Mematikan *Container*

No	Rute	Uji Coba	Hasil
1	/stats-server	VueJS mengambil data dari *middleware* lalu diolah dalam bentuk	OK.

Seperti terlihat pada gambar, ketika halaman diakses maka akan menampilkan penggunaan sumber daya CPU dan RAM yang dipakai oleh *server*. Data penggunaan yang ditampilkan berupa *timeseries*, semakin ke kanan semakin terbaru data yang ditampilkan 5.1.

Gambar 5.1: Dasbor Penggunaan *Resource Server*

5.3.1.2 Uji Coba *Middleware* Dapat Mengambil Data Penggunaan Sumber Daya *Container*

Uji coba ini dilakukan oleh *middleware*. Setiap container akan diambil data penggunaan sumber daya CPU, RAM, dan

LTA nya, lalu disimpan di *database*. Pada tabel `container`, *middleware* menyimpan *container* yang aktif saat ini di tabel `container`. Seperti terlihat pada gambar 5.3, setelah *middleware* menyimpan data *container* yang aktif saat ini, lalu mengambil data penggunaan sumber daya *container* karena pada *file* `config.ini` diatur pengambilan data setiap 10 menit, maka *middleware* akan menyimpan data penggunaan sumber daya CPU dan RAM yang dipakai oleh *server* ke *database*.

container_id	name	status	timestamps
3bf8074c8e	moodle3	running	2017-06-21 23:13:57
4f47052e28	moodle4	running	2017-06-21 23:13:57
50046a1067	moodle6	running	2017-06-21 23:13:57
589d330f9f	moodle7	running	2017-06-21 23:13:57
70d89a729c	moodle10	running	2017-06-21 23:13:57
8a4d55d21e	moodle9	running	2017-06-21 23:13:57
bf159344f6	moodle5	running	2017-06-21 23:13:57
c7791b6f3c	moodle1	running	2017-06-21 23:13:57
f360ec2998	moodle2	running	2017-06-21 23:13:57
f5fafe2c30	moodle8	running	2017-06-21 23:13:57

Gambar 5.2: Tabel `containers`

Gambar 5.3: *Database* Penggunaa *Resource Server*

Tabel 5.6: Uji Fungsionalitas Mendapatkan Data *Docker Container*

No	Moodle Container	Uji Coba	Hasil
1	Moodle Container 1	Middleware mengambil data sumber daya *server* dan menyimpannya ke MySQL.	*Middleware* berhasil mengambil data penggunaan sumber daya *middleware*
2	Moodle Container 2	Middleware mengambil data sumber daya *server* dan menyimpannya ke MySQL.	*Middleware* berhasil mengambil data penggunaan sumber daya *middleware*.

Tabel 5.6: Uji Fungsionalitas Mendapatkan Data *Docker Container*

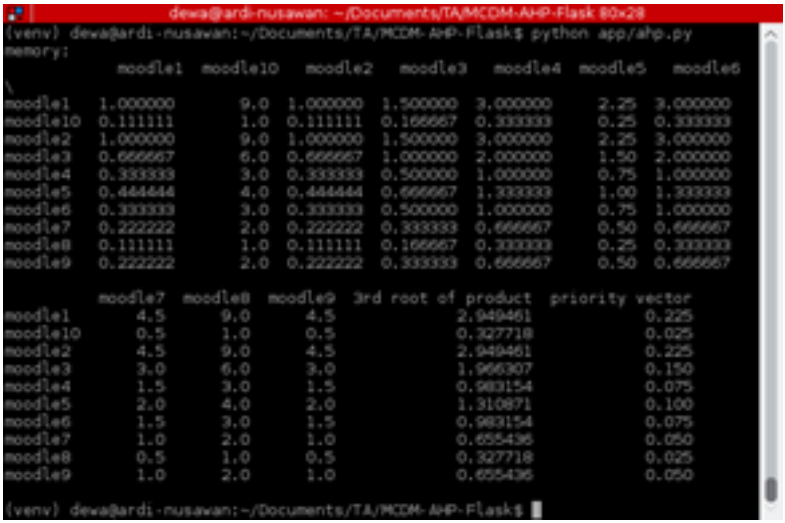
No	Moodle Container	Uji Coba	Hasil
3	Moodle Container 3	Middleware mengambil data sumber daya *server* dan menyimpannya ke MySQL.	*Middleware* berhasil mengambil data penggunaan sumber daya *middleware*.
4	Moodle Container 4	Middleware mengambil data sumber daya *server* dan menyimpannya ke MySQL.	*Middleware* berhasil mengambil data penggunaan sumber daya *middleware*.
5	Moodle Container 5	Middleware mengambil data sumber daya *server* dan menyimpannya ke MySQL.	*Middleware* berhasil mengambil data penggunaan sumber daya *middleware*.
6	Moodle Container 6	Middleware mengambil data sumber daya *server* dan menyimpannya ke MySQL.	*Middleware* berhasil mengambil data penggunaan sumber daya *middleware*.

Tabel 5.6: Uji Fungsionalitas Mendapatkan Data *Docker Container*

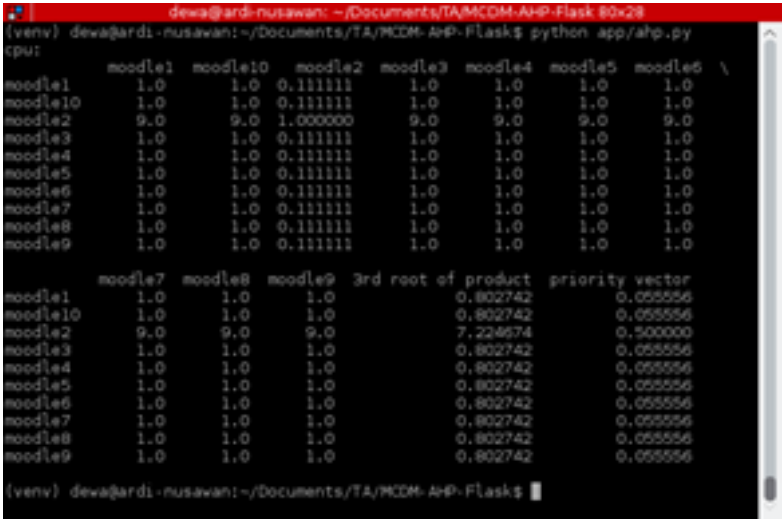
No	Moodle Container	Uji Coba	Hasil
7	Moodle Container 7	Middleware mengambil data sumber daya *server* dan menyimpannya ke MySQL.	*Middleware* berhasil mengambil data penggunaan sumber daya *middleware*.
8	Moodle Container 8	Middleware mengambil data sumber daya *server* dan menyimpannya ke MySQL.	*Middleware* berhasil mengambil data penggunaan sumber daya *middleware*.
9	Moodle Container 9	Middleware mengambil data sumber daya *server* dan menyimpannya ke MySQL.	*Middleware* berhasil mengambil data penggunaan sumber daya *middleware*.
10	Moodle Container 10	Middleware mengambil data sumber daya *server* dan menyimpannya ke MySQL.	*Middleware* berhasil mengambil data penggunaan sumber daya *middleware*.

5.3.2 Uji Perangkat Komputer Dapat Melakukan Pehitungan AHP Untuk Menentukan *Container* Mana Yang Akan Dimatikan

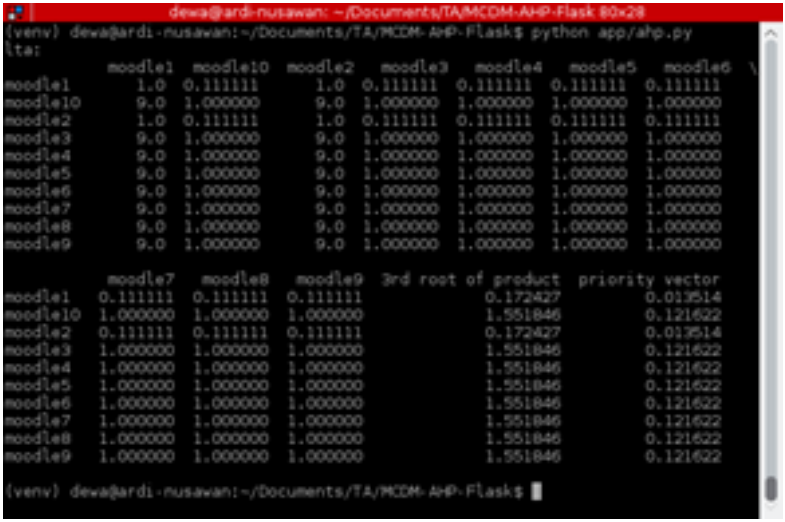
Uji coba dilakukan dengan menjalankan *middleware* selama 2 jam, 6 jam, dan 1 hari. *Middleware* dijalankan dengan mengeksekusi file `start.sh` yang isinya seperti pada *file* B.6. Setiap 10 menit, *middleware* akan menghitung nilai AHP dan menyimpannya di *database* pada tabel `result` seperti terlihat pada gambar 5.7. Apache JMeter mengakses setiap *container* Moodle yang tersedia, dari *container* 1 sampai dengan *container* 10.



Gambar 5.4: Perhitungan *Rating* Setiap *Container* Untuk *Memory* (RAM) Pada *Middleware*



Gambar 5.5: Perhitungan *Rating* Setiap *Container* Untuk CPU Pada *Middleware*



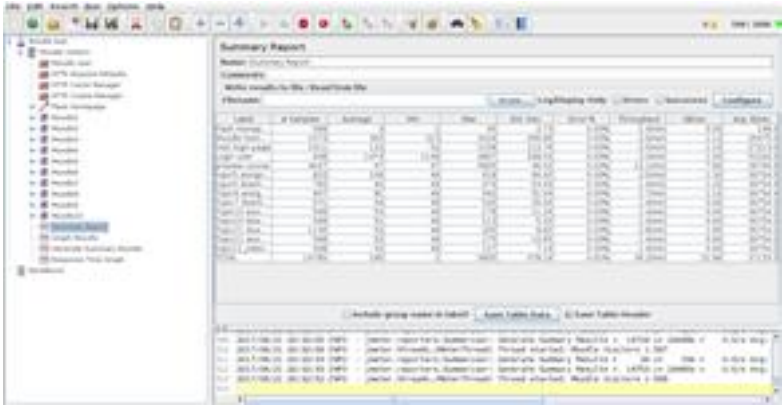
Gambar 5.6: Perhitungan *Rating* Setiap *Container* Untuk LTA Pada *Middleware*

Gambar 5.7: *Database* Penggunaa *Resource Server*

Selain dapat dilihat dari tabel `result`, pada *middleware* juga mengeluarkan output di console seperti pada gambar 5.8 yang menandakan bahwa *middleware* telah berhasil menghitung nilai AHP dan menyimpannya di *database*.

Gambar 5.8: *Console Log* Menghitung *AHP*

Pada gambar 5.9,terdapat `summary report` yang ditampilkan oleh JMeter, yang berisi berbagai informasi tentang halaman Moodle yang diakses. Nantinya informasi ini berguna untuk menganalisa pengaruh penggunaan algoritma AHP pada *container* Moodle.



Gambar 5.9: JMeter Mengakses Setiap *Container* Yang Tersedia

5.3.2.1 Uji Coba *Middleware* Dapat Menghentikan *Container*

Pada uji coba ini, proses *middleware* menghentikan *container* mana yang dihentikan dapat diperhatikan dengan membandingkan data pada tabel `containers` setiap 10 menit. Namun, setiap 10 menit belum tentu terdapat sebuah *container* yang dimatikan. Untuk itu, data yang ditampilkan pada gambar 5.10 adalah ketika *middleware* pertama kali mengambil data *container* yang aktif, dan gambar 5.11 adalah ketika middleware telah dijalankan selama 6 jam. Dapat dilihat, setelah 6 jam pada *server* hanya menyisakan 2 buah *container* yang aktif.

container_id	name	status	timestamps
3bf8074c8e	moodle3	running	2017-06-21 23:13:57
4f47052e28	moodle4	running	2017-06-21 23:13:57
50046a1067	moodle6	running	2017-06-21 23:13:57
589d330f9f	moodle7	running	2017-06-21 23:13:57
70d89a729c	moodle10	running	2017-06-21 23:13:57
8a4d55d21e	moodle9	running	2017-06-21 23:13:57
bf159344f6	moodle5	running	2017-06-21 23:13:57
c7791b6f3c	moodle1	running	2017-06-21 23:13:57
f360ec2998	moodle2	running	2017-06-21 23:13:57
f5fafe2c30	moodle8	running	2017-06-21 23:13:57

Gambar 5.10: Tabel `result` Awal

container_id	name	status	timestamps
c7791b6f3c	moodle1	running	2017-06-22 05:13:57
f360ec2998	moodle2	running	2017-06-22 05:13:57

Gambar 5.11: Tabel `result` Akhir

Tabel 5.7: Uji Coba Sistem Dapat Melakukan Perhitungan AHP

No	Fungsi	Uji Coba	Hasil
1	stats()	Memanggil fungsi `score()` pada file `ahp.py`	*Middleware* dapat memanggil fungsi `score()` pada file `ahp.py`
2	score()	Menghitung nilai AHP untuk setiap *container*	*Middleware* berhasil menghitung AHP dan menghasilkan *container* mana yang terpilih untuk dimatikan.

5.3.3 Uji Performa

Uji performa dilakukan dengan menjalankan aplikasi *middleware* pada server dengan berbagai durasi. Pada *middleware* dapat diatur aksi yang dilakukan setelah perhitungan AHP selesai, yakni *running* dan *stop*.

Tabel 5.8: Jumlah Akses Pengguna dan Durasi

No	Akses Pengguna	Durasi
1	500	2 Jam
2	500	6 Jam
3	2000	1 Hari

Hasil dari uji performa dapat dilihat seperti pada gambar 5.12. Terdapat 2 grafik, yakni penggunaan CPU (atas) dan RAM (bawah). Garis vertikal pada CPU menunjukkan persentase penggunaan dari 0 - 100 % sedangkan pada RAM menunjukkan banyaknya penggunaan RAM dalam satuan Byte dari 0 - maksimum penggunaan RAM. Pada garis horizontal, baik pada gambar CPU maupun RAM menunjukkan baris waktu data, semakin ke kanan maka data semakin terbaru. Panjang garis horizontal tergantung dari lamanya uji coba dilaksanakan.

5.3.3.1 Dengan AHP Uji Coba ke-1(RAM > LTA > CPU)

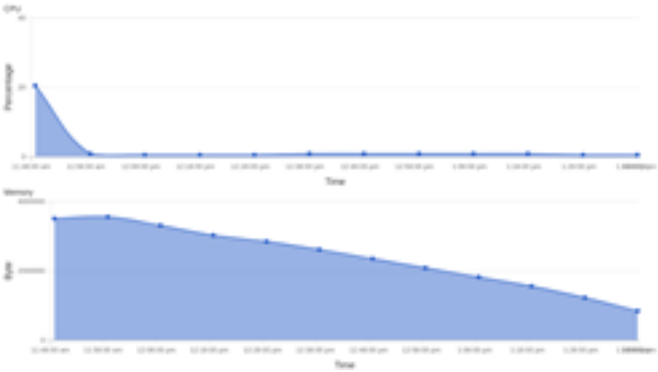
Pada uji coba, dijalankan *middleware* dengan AHP dan berdasarkan referensi dari pengguna yakni RAM lebih penting daripada LTA, dan LTA lebih penting daripada CPU, dan diberikan nilai sesuai pada tabel berikut:

Tabel 5.9: Jumlah Akses Pengguna dan Durasi

No	Perbandingan	Nilai
1	Memory/LTA	2

Tabel 5.9: Jumlah Akses Pengguna dan Durasi

No	Perbandingan	Nilai
2	Memory/CPU	4
3	LTA/CPU	2



Gambar 5.12: Hasil Uji Coba dengan AHP Selama 2 Jam

Gambar 5.13: Hasil Uji Coba dengan AHP Selama 6 Jam

Gambar 5.14: Hasil Uji Coba dengan AHP Selama 1 Hari

Dapat dilihat pada hasil uji coba, jika pada *middleware* dijalankan aksi *stop* setelah perhitungan AHP, maka terjadi pengurangan penggunaan sumber daya RAM. Namun, terdapat efek samping, yakni jika sebuah *container* sebelumnya telah dimatikan lalu karena terdapat akses ke *container* tersebut, terjadi *error* yang disebabkan proses menunggu dari keadaan *container stop* ke *running*. Namun *error* hanya terjadi pada akses yang pertama, akses kedua dan selanjutnya *container* akan dapat langsung diakses.

Dapat dilihat juga dari gambar diatas, semakin lama *middleware* dijalankan, *middleware* mampu menjaga penggunaan RAM dikisaran angka 2.9GB. Dibandingkan dengan tidak menggunakan *middleware* RAM yang berada pada kisaran 4 GB.

5.3.3.2 Dengan AHP Uji Coba ke-2 (CPU > Memory > LTA

Pada uji coba, dijalankan *middleware* dengan AHP dan berdasarkan referensi dari pengguna yakni CPU lebih penting daripada RAM, dan RAM lebih penting daripada LTA, dan diberikan nilai sesuai pada tabel berikut:

Tabel 5.10: Jumlah Akses Pengguna dan Durasi

No	Perbandingan	Nilai
1	Memory/LTA	2
2	CPU/Memory	2
3	CPU/LTA	4

Gambar 5.15: Hasil Uji Coba dengan AHP Selama 2 Jam

Gambar 5.16: Hasil Uji Coba dengan AHP Selama 6 Jam

Gambar 5.17: Hasil Uji Coba dengan AHP Selama 1 Hari

Dapat dilihat pada hasil uji coba, jika pada *middleware* dijalankan aksi *stop* setelah perhitungan AHP, maka terjadi pengurangan penggunaan sumber daya CPU. Namun, terdapat efek samping, yakni jika sebuah *container* sebelumnya telah dimatikan lalu karena terdapat akses ke *container* tersebut, terjadi *error* yang disebabkan proses menunggu dari keadaan *container stop* ke *running*. Namun *error* hanya terjadi pada akses yang pertama, akses kedua dan selanjutnya *container* akan dapat langsung diakses.

Dapat dilihat juga dari gambar diatas, penggunaan RAM pada uji coba selama 2 jam dan 6 jam tidak menunjukkan penurunan penggunaan. Ini disebabkan karena memang penggunaan RAM bukan prioritas.

5.3.3.3 Dengan AHP Uji Coba ke-3 (LTA > Memory > CPU

Pada uji coba, dijalankan *middleware* dengan AHP dan berdasarkan referensi dari pengguna yakni LTA lebih penting daripada RAM, dan RAM lebih penting daripada CPU, dan diberikan nilai sesuai pada tabel berikut:

Tabel 5.11: Jumlah Akses Pengguna dan Durasi

No	Perbandingan	Nilai
1	LTA/Memory	2
2	Memory/CPU	4
3	LTA/CPU	2

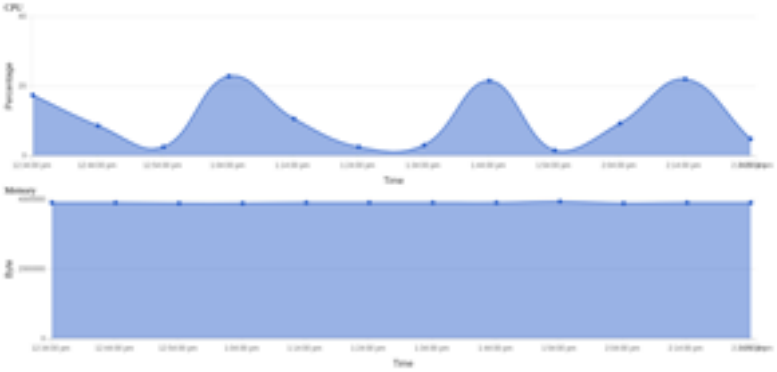
Gambar 5.18: Hasil Uji Coba dengan AHP Selama 2 Jam

Gambar 5.19: Hasil Uji Coba dengan AHP Selama 6 Jam

Gambar 5.20: Hasil Uji Coba dengan AHP Selama 1 Hari

Dapat dilihat juga dari gambar diatas, penggunaan CPU maupun RAM pada uji coba selama 2 jam dan 6 jam tidak menunjukkan penurunan penggunaan yang berarti. Ini disebabkan karena memang penggunaan CPU dan RAM bukan prioritas.

5.3.3.4 Tanpa AHP



Gambar 5.21: Hasil Uji Coba tanpa AHP Selama 2 Jam

Gambar 5.22: Hasil Uji Coba tanpa AHP Selama 6 Jam

Gambar 5.23: Hasil Uji Coba tanpa AHP Selama 1 Hari

Dapat dilihat dari gambar diatas, penggunaan CPU maupun RAM pada uji coba selama 2 jam, 6 jam dan 1 hari tidak menunjukkan penurunan penggunaan sama sekali karena memang tidak ada *container* yang dimatikan.

(Halaman ini sengaja dikosongkan)

BAB VI

PENUTUP

Bab ini membahas kesimpulan yang dapat diambil dari tujuan pembuatan sistem dan hubungannya dengan hasil uji coba dan evaluasi yang telah dilakukan. Selain itu, terdapat beberapa saran yang bisa dijadikan acuan untuk melakukan pengembangan dan penelitian lebih lanjut.

6.1 Kesimpulan

Dari proses perancangan, implementasi dan pengujian terhadap sistem, dapat diambil beberapa kesimpulan berikut:

1. Sistem dapat menentukan *container* yang akan dimatikan dengan menggunakan algoritma AHP dengan kriteria-kriteria seperti penggunaan CPU, RAM , dan waktu akhir diakses pada masing-masing *container* yang ada.
2. Dengan menggunakan AHP, dibandingkan dengan tidak menggunakan pengoptimalan penggunaan sumber daya, sistem dapat menentukan *container* yang akan dimatikan dengan efisien berdasarkan penggunaan CPU, RAM , dan waktu terakhir diakses pada masing-masing *container* yang ada, dimana dengan menggunakan AHP ketersediaan akhir memori dari setiap *container* lebih stabil dengan kisaran 2.9 GB, sedangkan tanpa pengoptimalan ketersediaan akhir memori tinggi karena seluruh *container* selalu menyala, dengan rentang yang cukup jauh yakni stabil di kisaran 4.0 GB.

6.2 Saran

Berikut beberapa saran yang diberikan untuk pengembangan lebih lanjut:

- Dalam perhitungan AHP, jika jumlah container banyak maka akan memakan waktu yang cukup lama. Optimasi perhitungan dibutuhkan agar sistem dapat lebih optimal.
- Untuk menyimpan data penggunaan sumber daya *server* dan *container*, lebih baik menggunakan database InfluxDB karena memang dikhususkan untuk penyimpanan data yang berbasis *timeseries*.

DAFTAR PUSTAKA

[1] G. Coyle, "The Analytic Hierarchy Process (AHP)," *Practical Strategy. Open Access Material. AHP*, no. Pearson Education Limited, 2004.

[2] K. Teknomo, "Analytic hierarchy process (AHP) tutorial," *Retrieved on January*, vol. 11, p. 2011, 2006.

[3] D. Bernstein, "Containers and Cloud: From LXC to Docker to Kubernetes," *IEEE Cloud Computing*, vol. 1, pp. 81–84, Sept. 2014.

[4] C. Boettiger, "An introduction to Docker for reproducible research, with examples from the R environment," *ACM SIGOPS Operating Systems Review*, vol. 49, pp. 71–79, Jan. 2015. arXiv: 1410.0846.

[5] python.org, "What is Python? Executive Summary," July 2017. https://www.python.org/doc/essays/blurb/.

[6] Armin Ronacher, "Welcome to Flask — Flask Documentation (0.12)," 2017. http://flask.pocoo.org/docs/0.12/.

[7] "MySQL." https://www.mysql.com/.

[8] "bitnami-docker-moodle: Bitnami Docker Image for Moodle," May 2017. https://github.com/bitnami/bitnami-docker-moodle.

[9] "middleware - definition of middleware in English | Oxford Dictionaries." https://en.oxforddictionaries.com/definition/middleware.

[10] Amandeep, F. Mohammad, and V. Yadav, "Automatic decision making for multi-criteria load balancing in cloud environment using AHP," in *Communication Automation International Conference on Computing*, pp. 569–576, May 2015.

(Halaman ini sengaja dikosongkan)

LAMPIRAN A

INSTALASI PERANGKAT LUNAK

1.1 Instalasi Lingkungan Docker

Proses pemasangan Docker dpat dilakukan sesuai tahap berikut:

1. Menambahkan repository Docker
   Langkah ini dilakukan untuk menambahkan *repository* Docker ke dalam paket `apt` agar dapat di unduh oleh Ubuntu. Untuk melakukannya, jalankan perintah berikut:

```
sudo apt-get -y install \
   apt-transport-https \
   ca-certificates \
   curl

curl -fsSL https://download.docker.com/linux/
    ubuntu/gpg | sudo apt-key add -

sudo add-apt-repository \
    "deb [arch=amd64] https://download.docker.com/
     linux/ubuntu \
    $ (lsb_release -cs) \
    stable"

sudo apt-get update
```

2. Mengunduh Docker
   Docker dikembangkan dalam dua versi, yaitu CE (*Community Edition*) dan EE (*Enterprise Edition*). Dalam pengembangan sistem ini, digunakan Docker CE karena merupakan versi Docker yang gratis. Untuk mengunduh Docker CE, jalankan perintah `sudo apt-get -y install docker-ce`.

3. Mencoba menjalankan Docker
   Untuk melakukan tes apakah Docker sudah terpasang dengan benar, gunakan perintah `sudo docker run hello-world`.

1.2 Instalasi Docker Compose

1. Memeriksa instalasi *docker compose* yang terbaru, dengan perintah berikut:

```
sudo curl -o /usr/local/bin/docker-compose -L\
   "https://github.com/docker/compose/releases\
   /download/1.11.2/docker-compose- \
   $(uname -s)-$(uname -m)" \
```

2. Mengeset *permission* agar *docker-compose* dapat dijalankan:

```
$ sudo chmod +x /usr/local/bin/docker-compose
```

3. Verifikasi instalasi sukses dilakukan dengan memeriksa versi:

```
$ docker-compose -v
```

1.3 Instalasi Pustaka Python

Dalam pengembangan sistem ini, digunakan berbagai pustaka pendukung. Pustaka pendukung yang digunakan merupakan pustaka untuk bahasa pemrograman Python. Berikut adalah daftar pustaka yang digunakan dan cara pemasangannya:

- Instalasi Python3 $ `sudo apt-get install python3`
- Instalasi Pip Python3 $ `sudo apt-get install python3-pip`

- Python Dev
  $ `sudo apt-get install python-dev`
- Flask
  $ `sudo pip install Flask`
- docker-py
  $ `sudo pip install docker`
- MySQLd
  $ `sudo apt-get install python-mysqldb`
- pandas
  $ `sudo pip install pandas`

1.4 Instalasi MySQL

Berikut langkah - langkah untuk menginstal *database* MySQL. :

1. Langkah 1 — Instalasi MySQL
   Pada Ubuntu 16.04, hanya versi terakhir MySQL yang terdapat pada paket repository APT secara *default*, yakni versi 5.7. Untuk menginstalnya,cukup *update index* paket APT pada komputer dan install paket default dengan apt-get.

   ```
   $ sudo apt-update
   $ sudo apt-get install mysql-server
   ```

2. Langkah 2 — Konfigurasi MySQL
   Pada instalasi yang baru, saya menginginkan untuk menajalankan *script* keamanan. Ini diperlukan untuk memastikan password aman dan merubah settingan database apakah bisa di*remote* atau tidak. Menjalankan *script* keamanan dilakukan dengan cara:

   ```
   $ sudo mysql_secure_installation
   ```

Perintah ini akan meminta kamu untuk memasukkan password *root*. Kamu bisa menekan Y lalu menekan ENTER untuk menggunakan setingan bawaan pada setiap pertanyaan, dengan pengucualian pada permintaan untuk merubah password. Kamu bisa mengganti ataupun tidak sesuai keinginan.

3. Langkah ke 3 — Testing MySQL Seharusnya MySQL sudah berjalan secara otomatis. Untuk mengetesnya, jalankan perintah ini:

```
$ systemctl status mysql.service
```

Jika hasil seperti gambar berikut, maka MySQL berhaisl dijalankan:

```
● mysql.service - MySQL Community Server
  Loaded: loaded (/lib/systemd/system/mysql.
  service; enabled; vendor preset: en
  Active: active (running) since Wed 2016-11-23
  21:21:25 UTC; 30min ago
  Main PID: 3754 (mysqld)
  Tasks: 28
  Memory: 142.3M
  CPU: 1.994s
  CGroup: /system.slice/mysql.service
  └─3754 /usr/sbin/mysqld
```

Jika MySQL tidak berjalan, bisa dijalankan dengan `sudo systemctl mysql start`. Untuk pengecekan tambahan, kamu bisa mencoba koneksi ke database dengan mysqladmin, yakni pengguna yang menjalankan perintah administratif. Sebagai contoh,perintah ini menjalankan MySQL sebagai root (-u root), menggunakan *password* (-p), dan menampilkan versi database.

```
$ mysqladmin -p -u root version
```

Jika instalasi sukses dan MySQL berjalan normal, maka hasilnya kurang lebih seperti berikut:

```
mysqladmin  Ver 8.42 Distrib 5.7.16, for Linux on
x86_64
Copyright (c) 2000, 2016, Oracle and/or its
affiliates. All rights reserved.

Oracle is a registered trademark of Oracle
Corporation and/or its affiliates. Other names
may be trademarks of their respective owners.

Server version      5.7.16-0ubuntu0.16.04.1
Protocol version    10
Connection      Localhost via UNIX socket
UNIX socket     /var/run/mysqld/mysqld.sock
Uptime:         30 min 54 sec

Threads: 1  Questions: 12  Slow queries: 0  Opens:
115  Flush tables: 1  Open tables: 34  Queries per
second avg: 0.006
```

1.5 Pemasangan Kerangka Kerja VueJS

Pada pengembangan sistem ini, penggunaan pustaka VueJS dibangun di atas konfigurasi VueJS CLI. Untuk memasang VueJS CLI, gunakan perintah `npm install -g vue-cli`. Setelah terpasang, untuk membangun aplikasinya jalankan perintah `ue init webpack my-project`. Setelah proses tersebut, dasar dari aplikasi sudah terbangun dan siap untuk dikembangkan lebih lanjut.

(Halaman ini sengaja dikosongkan)

LAMPIRAN B

KODE SUMBER

Modul AHP fungsi weight_of_criteria

Kode Sumber B.1: Modul AHP fungsi weight_of_criteria

```
def weight_of_criteria(*args, **kwargs):
    # read from cfg file
    kwargs["config"] = True
    parameter = database.select("parameter
        ", **kwargs)
    parameter = sorted(parameter.items(),
        key=lambda x: x[1])
    p_name = [x[1] for x in parameter]
    index = p_name
    columns = p_name
    type_data = ['float32'] * len(p_name)
    dtype = list(zip(p_name, type_data))
    values = np.zeros(len(p_name), dtype=
        dtype)
    data_matrix = pd.DataFrame(values,
        index=index, columns=columns)

    kwargs.clear()
    kwargs["config"] = True
    comparison = database.select("
        comparison", **kwargs)
    c_key = [x for x in list(comparison.
        keys())]
    for idx, row in enumerate(data_matrix.
        index.values):
         for idy, column in enumerate(
             data_matrix.columns.values):
              for i in range(len(c_key)):
```

```
                if c_key[i].split("/")[0]
                   == row.lower() and c_key
                   [i].split("/")[1] ==
                   column.lower():
                    data_matrix.loc[row,
                       column] = comparison
                       [c_key[i]]
                    data_matrix.loc[column,
                        row] = 1 / int(
                       comparison[c_key[i
                       ]])
                    break
            if column == row:
                data_matrix.loc[row, column
                   ] = 1
    data_matrix["3rd root of product"] =
       data_matrix.product(axis=1) ** (1 /
       len(parameter))
    data_matrix["priority vector"] =
       data_matrix["3rd root of product"] /
        data_matrix["3rd root of product"].
       sum()
    return data_matrix
```

Modul AHP fungsi rating_each_node

Kode Sumber B.2: Modul AHP fungsi rating_each_node

```
def rating_each_node(column_name, *args, **
   kwargs):
    name = column_name
    # print("{} rating".format(name))
```

```
# 1. mendapatkan semua data CPU, memory
    , LTA, cari terendah dan tertinggi,
    bagi 9 kolom
kwargs["column"] = "container_id,
    timestamps, name"
kwargs["where"] = "status = 'running'"
kwargs["sort"] = "name"
container = database.select("containers
    ", **kwargs)
if not container:
    return False, {"message": "no
        container active"}
c_id = '{}'.format([x[0] for x in
    container])
c_id = "".join(c_id)
c_id = c_id[1: 1]
ts = container[0][1]

# kwargs.clear()
kwargs["column"] = "container_id, {
    column}, container_name".format(
    column=name)
kwargs["sort"] = "container_name"
# data = database.select("stats", **
    kwargs)
if "hour" in kwargs.keys():
    kwargs["where"] = "container_id IN
        ({c_id}) AND timestamps BETWEEN
        '{hour_from}' and '{hour_to}'".
        format(
        c_id=c_id, hour_from=kwargs["
            hour_from"], hour_to=kwargs
            ["hour_to"])
```

```
    elif "day" in kwargs.keys():
        kwargs["where"] = "container_id IN
            ({c_id}) AND timestamps BETWEEN
            '{day_from}' and '{day_to}'".
            format(
             c_id=c_id, day_from=kwargs["
                day_from"], day_to=kwargs["
                day_to"])
    elif "week" in kwargs.keys():
        kwargs["where"] = "container_id IN
            ({c_id}) AND timestamps BETWEEN
            '{week_from}' and '{week_to}'".
            format(
             c_id=c_id, week_from=kwargs["
                week_from"], week_to=kwargs
                ["week_to"])
    else:
        kwargs["where"] = "container_id IN
            ({c_id}) AND timestamps = '{ts
            }'".format(c_id=c_id, ts=ts)
    data = database.select("stats", **
        kwargs)

    c = Counter(v[2] for v in data)

    data_average = list()
    for val in c:
        average = sum(v[1] for v in data if
            v[2] == val) / float(c[val])
        id = [v for i, v in enumerate(data)
            if v[2] == val]
        id = id[0][0]
```

```
        data_average.append((id, average,
            val))
    if data_average:
        data = data_average
    if not all(data):
        return False, {"message": "no stats
            "}
    data_min = min(data, key=lambda key:
        key[1])
    data_max = max(data, key=lambda key:
        key[1])
    if not data_min and not data_max:
        data_range = list(np.arange(
            data_min[1], data_max[1], (
            data_max[1]   data_min[1]) / 9))
    else:
        data_range = [0]
    # 2. untuk setiap data, cek data ada
        pada kolom mana, masukkan ke matrik
    c_name = [x[2] for x in container]
    index = c_name
    columns = c_name
    type_data = ['float32'] * len(c_name)
    dtype = list(zip(c_name, type_data))
    values = np.zeros(len(c_name), dtype=
        dtype)

    data_matrix = pd.DataFrame(values,
        index=index, columns=columns)
    for idx, row in enumerate(data_matrix.
        index.values):
        for idy, column in enumerate(
            data_matrix.columns.values):
```

```
            tx = [item for item in data if
               item[2] == row][0][1]
            # ty = tx
            ty = [item for item in data if
               item[2] == column][0][1]

            for idx, val in enumerate(
               data_range[1:len(data_range)
               ]):
                if (tx >= data_range[idx
                   1] and tx < data_range[
                   idx]):
                    tx = idx
                    break
                elif tx >= data_range[len(
                   data_range)   1]:
                    tx = len(data_range)
                    break

            for idx, val in enumerate(
               data_range[1:len(data_range)
               ]):
                if (ty >= data_range[idx
                   1] and ty < data_range[
                   idx]):
                    ty = idx
                    break
                elif ty >= data_range[len(
                   data_range)   1]:
                    ty = len(data_range)
                    break

            # 3. Bandingkan row dan column
```

```
            if ty == 0:
                tz = 1
            else:
                tz = tx / ty
            data_matrix.loc[row, column] =
               tz

    # 4. untuk setiap row, hitung nilai
       geomean dan eigenvector
    data_matrix["3rd root of product"] =
       data_matrix.product(axis=1) ** (1 /
       database.total_data("containers"))
    data_matrix["priority vector"] =
       data_matrix["3rd root of product"] /
        data_matrix["3rd root of product"].
       sum()
    return True, data_matrix
```

Modul AHP fungsi score

```
def score(**kwargs):

    CPU = rating_each_node("CPU", **kwargs)
    Memory = rating_each_node("Memory", **
       kwargs)
    LTA = rating_each_node("
       last_time_access_percentage", **
       kwargs)

    if CPU[0] != False and Memory[0] !=
       False and LTA[0] != False:
```

```
cpu_dot_wc = np.dot(
    weight_of_criteria()["priority
    vector"].iloc[weight_of_criteria
    ().index.get_loc("CPU")],
                        CPU[1]["
                            priority
                            vector"])
mem_dot_wc = np.dot(
    weight_of_criteria()["priority
    vector"].iloc[weight_of_criteria
    ().index.get_loc("Memory")],
                        Memory [1]["
                            priority
                            vector"])
lta_dot_wc = np.dot(
    weight_of_criteria()["priority
    vector"].iloc[weight_of_criteria
    ().index.get_loc("LTA")],
                        LTA[1]["
                            priority
                            vector"])

c_score = cpu_dot_wc + mem_dot_wc +
     lta_dot_wc
c_score = [x.item() for x in list(
    c_score)]
c_name = list(map(list, containers
    ("container_id")))
c_ts = list(map(list, containers("
    timestamps")))[0][0].strftime("%
    Y %m %d %H:%M:%S")
c_name = [x[0] for x in c_name]
```

```
        score_final = dict(zip(c_name,
            c_score))
        score_max = max(score_final, key=
            score_final.get)
        score_min = min(score_final, key=
            score_final.get)

        score_final = {"status": "success",
             "result": score_final, "max":
            str(score_max), "min": str(
            score_min), "ts": c_ts}
        return score_final
    else:
        error = CPU[1]
        return {"status": "error", "message
            ": error}
```

config.ini

```
[mysql]
host: 127.0.0.1
username: root
password: Asddsaa1
database: MCDM AHP
port: 3306

[parameter]
Param1: Memory
Param2: LTA
Param3: CPU

[comparison]
```

```
Memory/LTA: 2
Memory/CPU: 4
LTA/CPU: 2

[timedelta]
hour: 1
day: 2
week: 1

[prefered]
; do: running if dont wanna pause/stop
do: stop
by: hour

[interval]
minute: 10
```

Modul Utils fungsi stats

Kode Sumber B.5: utils.py - stats()

```
def stats(**kwargs):

   # if containers == database.total_data("
      containers") and ahp.score()["status
      "] != "error":
    if ahp.score()["status"] != "error":
        kwargs.clear()
        kwargs["params"] = "
            container_id_hours,
            container_id_days,
            container_id_weeks, score_hours,
             " \
```

```
                                ”score_days,
                                    score_weeks,
                                    hour_from,
                                    day_from,
                                    week_from,
                                    timestamps”
        hour = timedelta(now,”hour”)
        day = timedelta(now,”day”)
        week = timedelta(now,”week”)
        score_hour = ahp.score(**hour)
        score_day = ahp.score(**day)
        score_week = ahp.score(**week)
        kwargs[”value”] = ”’{max_hour}’, ’{
            max_day}’, ’{max_week}’, ’{
            score_hour}’, ’{score_days}’, ”
            \
                               ”’{score_weeks}’,
                                    ’{hour_from
                                   }’, ’{day_from
                                   }’, ’{
                                   week_from}’,
                                   ’{timestamps
                                   }’”\
             .format(max_hour=score_hour[”
                max”],
```

start.sh

Kode Sumber B.6: start.sh

```
source $PWD/venv/bin/activate
( cmdpid=$BASHPID; (sleep 86460; kill
   $cmdpid) & exec python3 -u $PWD/app/app.
```

```
    py )
```

BIODATA PENULIS

I Dewa Putu Ardi Nusawan, akrab dipanggil Wawan lahir pada tanggal 30 Oktober 1995 di Singaraja, Bali. Penulis merupakan seorang mahasiswa yang sedang menempuh studi di Jurusan Teknik Informatika Institut Teknologi Sepuluh Nopember. Memiliki hobi antara lain membaca novel dan traveling. Selama menempuh pendidikan di kampus, penulis juga aktif dalam organisasi kemahasiswaan, antara lain Staff Departemen Dalam Negeri Mahasiswa Himpunan Mahasiswa Teknik Computer-Informatika pada tahun ke-2. Pernah menjadi staff Biro Dana Schematics tahun 2014 dan 2015. Selain itu penulis pernah menjadi asisten dosen di mata kuliah Sistem Operasi. Saat ini penulis juga merupakan Kordinator Laboratorium Pemrograman I Informatika ITS.